Cincom

AD/ADVANTAGE

MANTIS Language OpenVMS/UNIX

P39-1310-00

AD/Advantage® MANTIS Language OpenVMS/UNIX

Publication Number P39-1310-00

Cincom Systems, Inc.
55 Merchant Street
Cincinnati, Ohio 45246-3732
U. S. A.

PHONE: (513) 612-2300
FAX: (513) 612-2000
WORLD WIDE WEB: http://www.cincom.com

Attention:

Some Cincom products, programs, or services referred to in this publication may not be available in all countries in which Cincom does business. Additionally, some Cincom products, programs, or services may not be available for all operating systems or all product releases. Contact your Cincom representative to be certain the items are available to you.

Release information for this manual

The *AD/Advantage MANTIS Language OpenVMS/UNIX*, P39-1310-00, is dated February 12, 2001. This document supports Release 2.8 of MANTIS.

We welcome your comments

We encourage critiques concerning the technical content and organization of this manual. Please take the survey provided with the online documentation at your convenience.

Cincom Technical Support for AD/Advantage

FAX:	(513) 612-2000 Attn: MANTIS Support
E-mail:	helpna@cincom.com
Phone:	1-800-727-3525
Mail:	Cincom Systems, Inc. Attn: MANTIS Support 55 Merchant Street Cincinnati, OH 45246-3732 U. S. A.

Contents

About this book

Using this document

MANTIS® is an application development system that consists of design facilities (for example, screens and files) and a programming language. This manual describes syntax information for the MANTIS language along with simple MANTIS code components of MANTIS.

Document organization

The information in this manual is organized as follows:

Chapter 1—Introduction
Provides an introduction to MANTIS, and information on the Logical Terminal Interface, keyboard operations and terminal functions, and how to sign on to MANTIS.

Chapter 2—MANTIS language conventions
Outlines language guidelines used by MANTIS.

Chapter 3—MANTIS programming language
Provides details on statements and commands used in the program code.

Chapter 4—IBM compatibility considerations
Describes compatibility issues between MANTIS for VMS/UNIX and MANTIS for the Mainframe.

Chapter 5—Programming techniques
Offers suggestions for using MANTIS Logical Terminal Interface and external DO.

Appendix A—Status functions
Lists and describes indicators returned by MANTIS functions.

Appendix B—DBCS support feature

Describes the DBCS support feature for Japanese-language users.

Index

Revisions to this manual

The following changes have been made for this release:

- Updated Publications Release number from P25-1310-08 to P39-1310-00.
- Updated other Publication Titles and release numbers as necessary.
- Updated Main MANTIS Screen in “Signing on to MANTIS” on page 36.
- Updated Facility Selection Screen in “Signing on to MANTIS” on page 37.
- Updated “FORMAT” on page 217.
- Updated “SLICE” on page 342.
- Updated “SLOT” on page 343.

Conventions

The following table describes the conventions used in this document series:

Convention	Description	Example
Constant width type	Represents screen images and segments of code.	Screen Design Facility GET NAME LAST INSERT ADDRESS
Slashed b (ƀ)	Indicates a space (blank). The example indicates that a password can have a trailing blank.	WRITEPASSƀ
Brackets [ ]	Indicate optional selection of parameters. (Do not attempt to enter brackets or to stack parameters.) Brackets indicate one of the following situations.	
	A single item enclosed by brackets indicates that the item is optional and can be omitted. The example indicates that you can optionally enter a program name.	COMPOSE [*program-name*]
	Stacked items enclosed by brackets represent optional alternatives, one of which can be selected. The example indicates that you can optionally enter NEXT, PRIOR, FIRST, or LAST. (NEXT is underlined to indicate that it is the default.)	$\left[\begin{array}{l}\underline{\textbf{NEXT}}\\ \textbf{PRIOR}\\ \textbf{FIRST}\\ \textbf{LAST}\end{array}\right]$
Braces { }	Indicate selection of parameters. (Do not attempt to enter braces or to stack parameters.) Braces surrounding stacked items represent alternatives, one of which you must select. The example indicates that you must enter FIRST, LAST, or values for begin.	$\left\{\begin{array}{l}\textbf{FIRST}\\ \textbf{\textit{begin}}\\ \textbf{LAST}\end{array}\right\}$

Convention	Description	Example
<u>Underlining</u> (In syntax)	Indicates the default value supplied when you omit a parameter. The example indicates that if you do not specify ON, OFF, or a row and column destination, the system defaults to ON.	$\textbf{SCROLL}\begin{bmatrix}\underline{\textbf{ON}}\\ \textbf{OFF}\\ [\textbf{\textit{row}}][\textbf{,\textit{col}}]\end{bmatrix}$
	Underlining also indicates an allowable abbreviation or the shortest truncation allowed. The example indicates that you can enter either PRO or PROTECTED.	<u>PRO</u>TECTED
Ellipsis points...	Indicate that the preceding item can be repeated. The example indicates that you can enter (A), (A,B), (A,B,C), or some other argument in the same pattern.	(*argument*,...)
UPPERCASE	Indicates MANTIS reserved words. You must enter them exactly as they appear. The example indicates that you must enter CONVERSE exactly as it appears.	CONVERSE *name*
Italics	Indicate variables you replace with a value, a column name, a file name, and so on. The example indicates that you can supply a name for the program.	COMPOSE [*program-name*]
Punctuation marks	Indicate required syntax that you must code exactly as presented. ( ) parentheses . period , comma : colon ' single quotation marks	$[\textbf{LET}]v\begin{bmatrix}(i)\\(i,j)\end{bmatrix}[\textbf{ROUNDED}(n)] = e1\ [,e2,e3\ldots]$

Convention	Description	Example
UNIX OpenVMS	Information specific to a certain operating system is flagged by a symbol in a shadowed box (for example, UNIX) indicating which operating system is being discussed. Skip any information that does not pertain to your environment.	UNIX DBA will run on any terminal that supports the cursor library.

MANTIS documentation series

MANTIS is an application development system designed to increase productivity in all areas of application development, from initial design through production and maintenance. MANTIS is part of AD/Advantage®, which offers additional tools for application development. Below are listed the manuals offered with MANTIS in the OpenVMS™ and UNIX® environments, organized by task. You may not have all the manuals that are listed here. For a synopsis of each manual, refer to the *AD/Advantage MANTIS Application Development Tutorial OpenVMS/UNIX*, P39-1340.

Getting started

- *AD/Advantage MANTIS 2.8.x Installation and Startup OpenVMS/UNIX*, P39-0027*

General use

- *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300
- AD/Advantage MANTIS Language OpenVMS/UNIX, P39-1310
- *AD/Advantage MANTIS Messages and Codes OpenVMS/UNIX*, P39-1330*
- *AD/Advantage MANTIS Application Development Tutorial OpenVMS/UNIX*, P39-1340
- *AD/Advantage MANTIS SUPRA SQL Programming OpenVMS/UNIX*, P39-1345
- *AD/Advantage MANTIS Rdb Programming UNIX*, P39-1350
- *AD/Advantage MANTIS Oracle Programming UNIX*, P39-1355

NOTE

Manuals marked with an asterisk (*) are listed twice because you use them for different tasks.

Master User tasks

- *AD/Advantage MANTIS Administration OpenVMS/UNIX*, P39-1320
- *AD/Advantage MANTIS Messages and Codes OpenVMS/UNIX*, P39-1330*
- *AD/Advantage MANTIS 2.8.x Installation and Startup OpenVMS/UNIX*, P39-0027

NOTE

Manuals marked with an asterisk (*) are listed twice because you use them for different tasks.

Educational material

MANTIS educational material is available from your regional Cincom education department.

1

Introduction

MANTIS allows users with diverse backgrounds to solve their application development problems simply and quickly using a display terminal. It accomplishes this by removing the necessity of coding sheets, job control statements, source decks, and, most importantly, the waiting usually associated with the development and implementation of a new program or system.

MANTIS is composed of facilities (for creating screens, files, and other entities) and a programming language. By using MANTIS, you can:

- Design and create formatted screens to enable full use of the facilities available on today's computers to display data attractively.
- Design and create permanent files for storing and manipulating data.
- Create and test programs interactively. MANTIS also provides a compatibility mode that enables you to create programs that are compatible with MANTIS for the IBM mainframe. This mode protects you from executing statements and features that are not supported by MANTIS for the IBM mainframe. "IBM compatibility considerations" on page 403 provides considerations for using MANTIS statements (and associated designs) in compatibility mode.

MANTIS is extensively parameterized by a set of programmable options called MANTIS Options. A MANTIS Option either modifies the behavior of some aspect of the MANTIS language or restricts its capability in some way. For example, the IBM MANTIS Option controls whether MANTIS executes in IBM compatibility mode. This option can be turned on and off in a MANTIS program using the "SET $OPTION" (page 328) statement described in "MANTIS programming language" on page 89. Most MANTIS Options can be set when MANTIS starts up, by use of an Options Definition File, and/or by command line parameters. For more information about specifying MANTIS Options refer to *AD/Advantage MANTIS Administration OpenVMS/UNIX*, P39-1320.

This manual discusses the MANTIS programming language that you use to write your applications. It also explains the conventions of the MANTIS language and how MANTIS handles text and numeric data. (For information on the Program Design Facility and editing commands you use to modify your programs, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.)

The rest of this chapter provides information on the Logical Terminal Interface, on keyboard operations and terminal functions, and describes signing on to MANTIS.

What you see

MANTIS performs terminal I/O through the Logical Terminal Interface. This interface supports a logical display area with a maximum size of 32767 rows by 32767 columns. You can use MANTIS Screen Design to create a preformatted screen for your application, then use Program Design to write a MANTIS program to display your screen. Or, you can simply use Program Design to write a program which creates a non-preformatted display.

In either case, the Logical Terminal Interface allows you to use the full logical display area. Your physical screen acts as a window through which you can look at the contents of the logical display. The size of your window in rows and columns depends on the type and mode of your terminal.

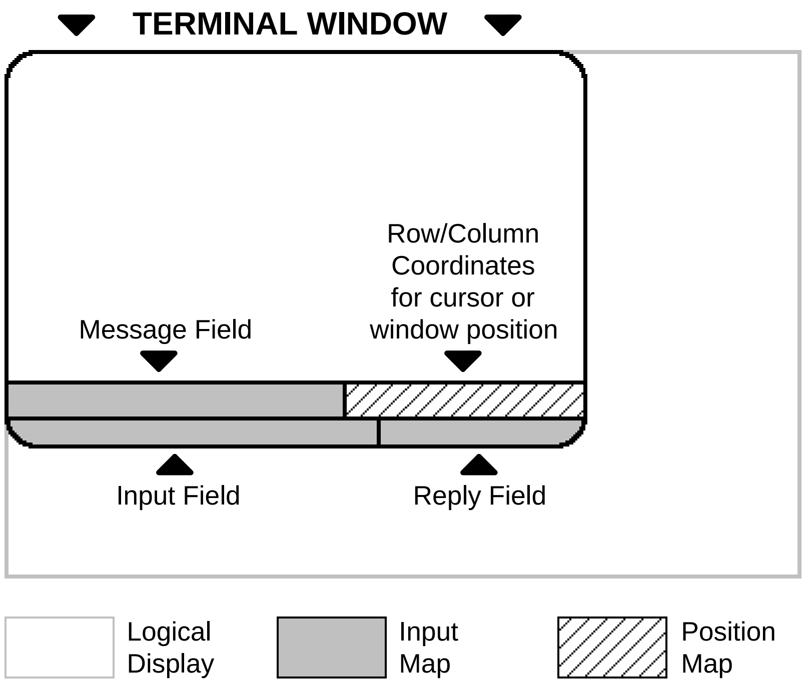
In addition, MANTIS can add floating maps to your display. A floating map has a constant horizontal and/or vertical position relative to the window in the logical display. When you move the window, the floating map moves (“floats”) in the same direction, retaining its horizontal and/or vertical position on the screen. MANTIS uses floating maps to provide display features described in the tables later in this section.

The two methods you can use in your MANTIS program to display data on the logical display are described on the following pages and involve a brief explanation of the Screen Design Facility and the CONVERSE, SHOW, OBTAIN, and HEAD statements. For more information on the Screen Design Facility refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300. For more details of these programming statements, see “MANTIS programming language” on page 89. If you are new to the MANTIS language, you may want to skip this section until you become more familiar with MANTIS Screen Design and these statements.

Full-screen mode

Full-screen mode describes the process that occurs when MANTIS uses the CONVERSE statement to display a screen (or “map”). To start this process, you must first create your screen design, defining heading and data fields and their attributes using the MANTIS Screen Design Facility. Then, you can display the map by using the CONVERSE statement in your MANTIS program. CONVERSE allows you to identify a particular map and its position on the logical display. When you run your program, MANTIS “converses” your map and waits for you to inspect the map and enter data from the keyboard. (Data that you enter is processed by MANTIS and moved to program variables to make it available for subsequent program statements.)

By default, MANTIS adds a floating map (called the *input map*) to the bottom two lines of the screen. The fields in these two lines are illustrated and described in the following figure and include fields for messages and user-entered data. These two lines may not appear on your “conversed” map if you have specified special attributes in your screen design which tell MANTIS to display the map over the last two lines (Full Display attribute) or tell MANTIS to protect the fields in the bottom line (Protect Bottom Line attribute). For more information about both attributes, refer to the *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 publication.



You can also use function keys (see “MANTIS terminal functions” on page 32) to add or remove another floating map (called the *position map*). This map occupies the last twelve characters of the second last line of the screen and contains two fields used to display row and column coordinates for the cursor or top left-hand corner of the window position. The fields on the full-screen mode display are listed and described in the following table.

Field	**Length**	**Function**
Message Field (*input map*)	Terminal width, less 12 columns	Displays system messages.
Row/Column Coordinates (*position map*)	5 columns for each	Two modes are available through function keys (see “MANTIS terminal functions” on page 32). MANTIS can either display the position of the cursor in the logical display or the row and column coordinates of the upper left corner of the window. When the position of the window is displayed, you can enter new values into the row and column fields. When you press RETURN, MANTIS interprets the values as follows: **unsigned value**—row or column number in the logical display **signed value**—row or column number relative to the current value **‘I’ precedes value**—row or column increment for window movement keys
Input Field (*input map*)	Terminal width, less 8 columns	Provides a field for entering input. Your MANTIS program picks up data from this field with the OBTAIN statement.
Reply Field (*input map*)	7 columns	Provides a field for entering function key values—for example, entering CANCEL in this field instead of pressing the CANCEL key. Your MANTIS program using the KEY function obtains the name of the key sequence entered here. The screen-name function returns the value of the Reply Field.

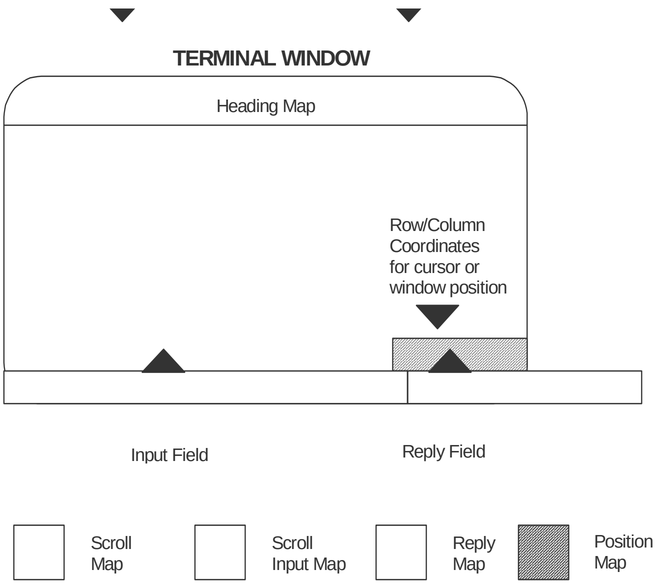
The CONVERSE statement also allows you to display more than one map on the logical display. A collection of maps on the logical display is called a map set. The order in which maps are added to the map set (that is, through multiple CONVERSE statements in your program) determines their order in the display as well as whether they can have their fields updated in the display. The logical display itself grows and shrinks according to the non-floating maps that you place in the map set.

Scroll mode

Scroll mode display describes the second method a MANTIS program can use to display data on the logical display. In this case, MANTIS uses a logical display area or “scroll map” for displaying data from SHOW, OBTAIN, and HEAD statements. The default size of the scroll map (and, as a result, the logical display) is 88 rows by 132 columns, but setting the SCRLROWS and SCRLCOLS MANTIS Options at MANTIS startup can modify the size. Each row in the display contains one line of output. Each new line of data is displayed at the bottom of the scroll map, with previous lines scrolling up the scroll map, one line at a time. You can use function keys (see “MANTIS terminal functions” on page 32) to move your terminal window so you can view other parts of the scroll map in the logical display.

MANTIS provides floating maps for the top two lines and bottom line of your display. The first two lines contain a *heading map*: the first line contains a centered heading (specified by a HEAD statement in the MANTIS program) and the second line is blank and simply separates the heading from the scroll-mode output that follows. In addition, MANTIS also uses a scroll-*input map* to display the Input Field for user-entered data on the bottom line of the screen. Input into this field must be solicited by the OBTAIN statement. By default, MANTIS also displays a reply map in the bottom right corner of the screen. As in full-screen mode, you can use function keys to add or remove a *position map* to display a row and column position for the cursor or the top, left corner of the window. You can also use GOLD/I to add or remove the Reply Field.

In your MANTIS program, the SHOW statement allows you to specify the data you want to display on your physical screen. An OBTAIN statement typically follows a SHOW statement in a MANTIS program to get your response to the SHOW data. Your MANTIS program with the KEY built-in function typically obtains data in the Reply Field.



Field	Length	Function
Input Field (*scroll input map*)	Same number of columns as your scroll map, less the columns for the Reply Field, if present	Provides a field for entering data (in this case, response to WAIT and OBTAIN).
Reply Field (*reply map*)	7 columns	Provides a field for entering function key values—for example, entering CANCEL in this field instead of performing the CANCEL terminal function. This value is obtained by your MANTIS program using the KEY built-in function or as a result of the CONVERSE statement.
Row/Column Coordinates (*position map*)	5 columns for each	Two modes are available through function keys. MANTIS can either display the position of the cursor or the row and column coordinates of the upper left corner of the window. When the position of the window is displayed, you can enter new values in the row and column fields. When you press RETURN, MANTIS interprets the values as follows: **unsigned value**—row or column number in the logical display **signed value**—row or column number relative to the current value **'I' precedes value**—row or column increment for window movement keys

Note that although your physical screen is conceptually a window on the scroll map, some data that is displayed may not be part of the scroll map. Output may be directed to your terminal by the system rather than MANTIS—messages from other users, for example. You can always refresh your terminal display by pressing CTRL-R.

Windowing

Your window on the logical display is the entire screen of your video display terminal. You can move this window around the logical display using the function keys listed in "MANTIS terminal functions" on page 32. When you move your window, your screen is updated to show the contents of the logical display at the new window position. You may notice, however, that some parts of your screen remain unchanged. This can happen for one of two reasons:

- To maintain compatibility with MANTIS for the IBM mainframe, MANTIS uses a field separator to simulate the invisible attribute byte that precedes every field on an IBM 3270-style terminal. When you design a map using the Screen Design Facility, one position in front of every field is reserved by default for the field separator. As a result, column one in the display (no matter where you move the window) is always blank for fields designed with field separators. For maps designed without field separators, column one is not reserved. For more information on specifying field separators, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 publication.

- MANTIS can modify the screen layout by adding the floating maps that were described on the previous pages.

Keyboard operation

MANTIS terminal I/O is designed for a standard terminal keyboard, consisting of a main keypad, an auxiliary keypad, cursor movement (arrow) keys, and at least four function keys.

The main keypad is normally used for data entry of alphanumeric, tabbing between screen input fields, and other functions. It usually includes a key labeled RETURN, which ends a screen input (CONVERSE).

The auxiliary keypad is normally used in function keypad mode (also called application keypad mode) to control terminal functions and interaction with the MANTIS application program. Numeric keypad mode enables the numeric keys in the auxiliary keypad to be interpreted as data entry keys rather than terminal functions.

The function keys are usually labeled F1, F2,... or, PF1, PF2,... and are used to either end a screen input (CONVERSE), or to modify a terminal function.

Refer to *AD/Advantage MANTIS Administration OpenVMS/UNIX*, P39-1320, for more detailed information about keyboard interpretation, and how you can effectively customize your keyboard for MANTIS.

Enter data

You can enter data only when the cursor is positioned in an unprotected field on the screen. When you press a data-entry key, the corresponding character is entered into the field at the cursor position, and the cursor moves one position to the right. If you are in overtype mode (the default mode), each character that you enter overwrites the character at the cursor position. In insert mode (turned on and off by the INSERT terminal function), each character that you enter is inserted at the cursor position and the following characters in the field are moved one position to the right.

Perform a terminal function

When you press one (or more) of the keys listed in "MANTIS terminal functions" on page 32, MANTIS performs a function that has an immediate effect on your terminal (for example, moving the window or initiating insert mode). It does not terminate a CONVERSE or an OBTAIN.

Perform a program function

When you press one (or more) of the keys listed in “MANTIS program functions” on page 35, MANTIS returns control to the MANTIS program running on your terminal. The program obtains the name of the program function that you selected (see “KEY” on page 269 and “screen-name” on page 324). The response to the program function key is determined by the logic of the MANTIS program. The names of the program functions are listed in “MANTIS program functions” on page 35. You can also enter any value in the Reply Field before pressing RETURN, and this value is returned to the MANTIS program as the program function name (as if the associated key had been pressed). Entering KILL in the Reply Field, however, terminates the MANTIS program.

In “MANTIS terminal functions” on page 32 and “MANTIS program functions” on page 35, each key is represented by the label that appears on its key cap, except that MANTIS follows the convention of using GOLD to refer to the key labeled PF1. When a key in the main keypad has the same label as a key in the numeric keypad, either key may be used unless otherwise noted. Key sequences always commence with GOLD or PF2 (also known as BLUE). Key sequences not mentioned in these tables are ignored. In a key sequence, a slash represents the pressing of one key after another. For example, single keys not assigned a function in the tables are data-entry keys. Be careful not to confuse a function key label (such as PF2) with the name of a program function (such as “PF2,” a MANTIS text value).

MANTIS terminal functions

Note	Default key sequence	Terminal function
Numeric	TAB ENTER (kent)	Move the cursor to the start of the next unprotected field.
VT	BACKSPACE F12	Move the cursor to the start of the current unprotected field (or to the start of the previous unprotected field if already at the start of the current field).
VT	LINEFEED F13 CTRL-E	Move the cursor past the last data character in the current unprotected field (or to the next unprotected field if already past the last data character).
	$\boxed{\rightarrow}$	Move the cursor right one column.
	$\boxed{\leftarrow}$	Move the cursor left one column.
	$\boxed{\uparrow}$	Move the cursor up one row.
	$\boxed{\downarrow}$	Move the cursor down one row.
	DELETE	Delete the data character to the left of the cursor.
	CTRL-C [SIGINT]	Force a program fault (FAULT 320) to occur in programming mode or abort the program MANTIS is currently executing.
	CTRL-R CTRL-W	Refresh the display by redisplaying the current screen.
	CTRL-U	Delete the characters in the field left of the cursor.
	CTRL-Y [SIGQUIT]	Abort MANTIS and return control to the system.
	GOLD=PF1 (kf1)	Initiates a 2-key sequence for terminal and program functions.
	PF2 (kf2)	Initiates a 3-key sequence for program functions only.
	PF3 (kf3)	Select insert mode.
	GOLD/PF3	Select overtype mode.
	CTRL-A	Alternate temporarily between insert and overtype modes.
	PF4 (kf4)	Delete the data character under the cursor.

Note	Default key sequence	Terminal function
	GOLD/PF4	Delete data characters from the cursor to the end of the field.
	GOLD/F	Select function keypad state.
	GOLD/N	Select numeric keypad state.
	GOLD/I	In full-screen mode, add or remove the *input map* in the bottom two lines of the screen. In scroll mode, add or remove the *reply map* in the last seven characters of the bottom line of the screen.
	GOLD/V	Displays extended edit attributes for a particular field. Position cursor over the field and press GOLD/V any time during data entry to the screen.
	GOLD/W	Add or remove the window *position map* in the last twelve characters of the second last line of the screen.
	GOLD/C	Add or remove the cursor *position map* in the last twelve characters of the second last line of the screen.
Function	KP8 (kri)	Move the window up by the row increment value.
Function	KP2 (kind)	Move the window down by the row increment value.
Function	KP4 (kpp)	Move the window left by the column increment value.
Function	KP6 (knp)	Move the window right by the column increment value.
Function	KP7 (ka1)	Move the window to the top left of the logical display.
Function	KP9 (ka3)	Move the window to the top right of the logical display.
Function	KP1 (kc1)	Move the window to the bottom left of the logical display.
Function	KP3 (kc3)	Move the window to the bottom right of the logical display.
Function	KP5 (kb2)	Display the entire scroll output map by scrolling from top to bottom.
Function	KP0	Copy the line under the cursor from the scroll output map to the *scroll input map*.

Note	Default key sequence	Terminal function
Function	MINUS (kprv) CTRL-B	Copy the previous input line from the scroll output map to the scroll *input map* or select box drawing mode in screen design.
Function	COMMA (knxt)	Copy the next input line from the scroll output map to the scroll *input map* or select box erasing mode in screen design.
Function	CTRL-K	Redefine the keyboard. For more details, refer to *AD/Advantage MANTIS Administration OpenVMS/UNIX*, P39-1320.
Function	RETURN ENTER (kent) GOLD/ENTER	“ENTER”
	GOLD/- CTRL-Z	“CANCEL”*

* You can also use the CANCEL function key sequence to terminate multipage scroll-mode output operations, returning a CANCEL to your MANTIS program.

Note legend

Numeric	in numeric keypad state	
Function	in function keypad state, on the auxiliary keypad only	
	VT	on VT200 or VT300 terminals
	( )	UNIX TERMINFO Capname
	[ ]	UNIX Signal trapped

MANTIS program functions

Note	Default key sequence	Terminal function
	GOLD/,	"PA1"
	GOLD/0	"PF0"
	GOLD/1	"PF1"
	GOLD/2	"PF2" (also for "PF3" to "PF9")
	PF2 (kf2)	Initiates a 3-key sequence
	PF2/0/0	"PF00"
	PF2/1/0	"PF10"
	PF2/1/1	"PF11" (also for "PF12" to "PF99")
VT	GOLD/E F20	"EDIT"
VT	GOLD/H HELP	"HELP"
	GOLD/K	"KILL"
	GOLD/Q	"QUIT"

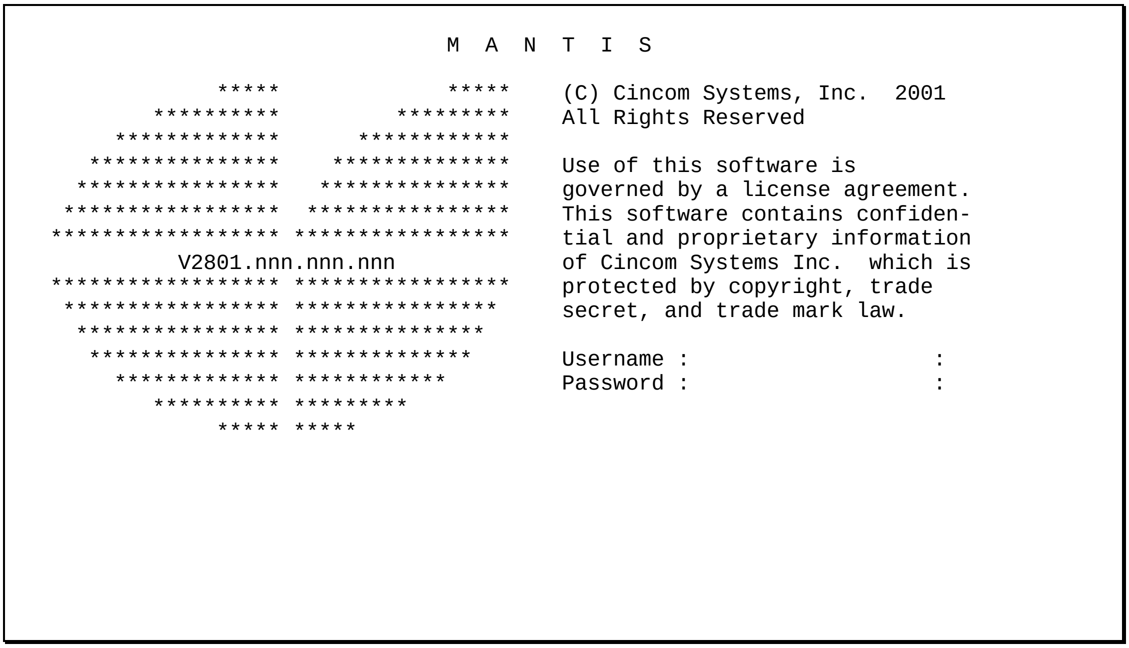
Note legend

Function	in function keypad state, on the auxiliary keypad only	
	VT	on VT200 or VT300 terminals
	( )	UNIX TERMINFO Capname

If the Reply Field is nonblank, any one of these key sequences generates the program function whose value is the value of the Reply Field.

Signing on to MANTIS

Every MANTIS installation has a person designated as the Master User. The Master User has access to certain facilities and information not available to all MANTIS users. Before signing on to MANTIS for the first time, obtain your MANTIS user name and password from your Master User. When you enter ‘MANTIS’ on your terminal, the sign-on screen appears as shown below (unless your Master User has changed the Cincom logo to another display).



```
                               M  A  N  T  I  S

             *****             *****    (C) Cincom Systems, Inc.  2001
        **********         *********    All Rights Reserved
     *************      ************
   ***************    **************    Use of this software is
  ****************   ***************    governed by a license agreement.
 *****************  ****************    This software contains confiden-
****************** *****************    tial and proprietary information
          V2801.nnn.nnn.nnn             of Cincom Systems Inc.  which is
****************** *****************    protected by copyright, trade
 ***************** ****************     secret, and trade mark law.
  **************** ***************
   *************** **************       Username :                   :
     ************* ************         Password :                   :
        ********** *********
             ***** *****
```

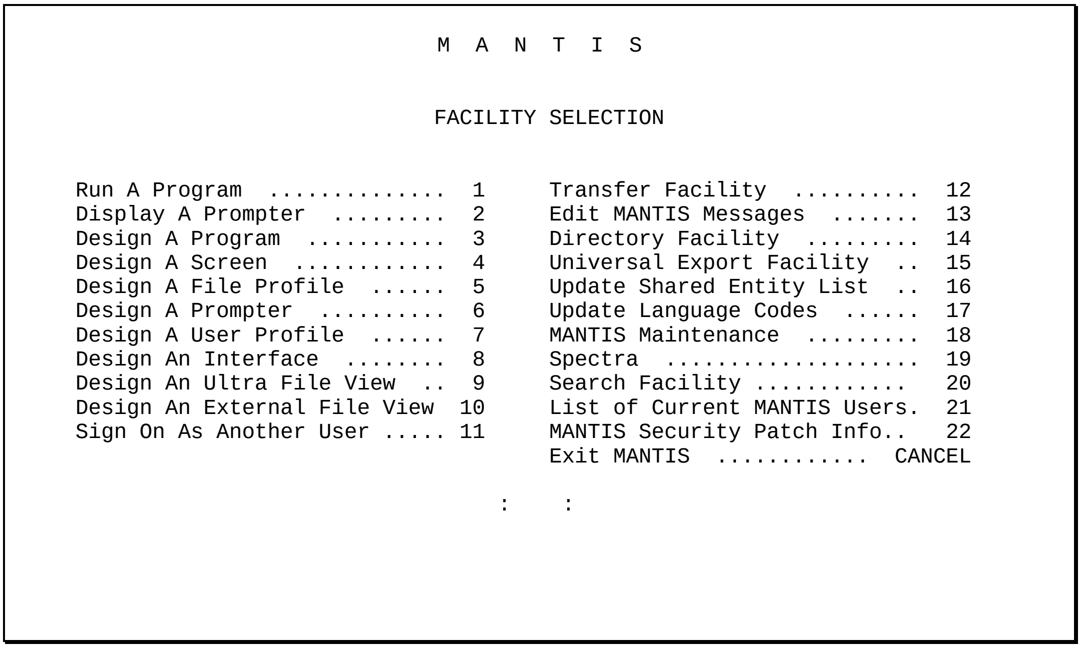
Your password must be entered exactly as your Master User specified it. Uppercase and lowercase letters are important. If the Cincom Logo screen is returned to you with the following message:

```
"All Developer License Seats are currently being used.  Please try
later"
```

This indicates that all current User Seat Licenses are being used at this time. Contact your MANTIS Administrator.

Enter your user name and password and press RETURN. Your Facility Selection menu appears and you are signed on to MANTIS.

The following screen illustration shows the standard facilities provided with MANTIS. Your Master User may have omitted some of these facilities or added new facilities to meet the needs of your installation.



```
                                  M  A  N  T  I  S


                                 FACILITY SELECTION


     Run A Program  ..............  1     Transfer Facility  ..........  12
     Display A Prompter  .........  2     Edit MANTIS Messages  .......  13
     Design A Program  ...........  3     Directory Facility  .........  14
     Design A Screen  ............  4     Universal Export Facility  ..  15
     Design A File Profile  ......  5     Update Shared Entity List  ..  16
     Design A Prompter  ..........  6     Update Language Codes  ......  17
     Design A User Profile  ......  7     MANTIS Maintenance  .........  18
     Design An Interface  ........  8     Spectra  ....................  19
     Design An Ultra File View  ..  9     Search Facility ............   20
     Design An External File View  10     List of Current MANTIS Users.  21
     Sign On As Another User ..... 11     MANTIS Security Patch Info..   22
                                          Exit MANTIS  ............  CANCEL

                                      :    :
```

To request a facility from the menu, either enter the number corresponding to the facility you want in the action field (: :) and press RETURN, or enter the key sequence for the correspondingly numbered program function. Remember that CANCEL corresponds to the GOLD/- key sequence.

The MANTIS programming language and conventions for using it are described in the following chapters.

NOTE

If you select a Developer type of MANTIS Facility such as “Design a Program”, “Design a Screen”, “Design a Interface” or any of the other Developer types of MANTIS Facilities you are considered a MANTIS Developer User. In order to access a Developer Facility, a Developer Seat License must be free.

If all Developer License Seats are currently being used, the MANTIS Screen below will appear. Please hit the ENTER Key to return to the Facility Selection Menu.

Compatibility mode

This manual describes a version of MANTIS where there is a high degree of compatibility between this version and the MANTIS for the IBM mainframe version. In fact, you can use the Universal Export Facility to move MANTIS programs—MANTIS files, screen designs, and other entities that you create on MANTIS—to MANTIS for the IBM mainframe. This allows application development to be shared between minicomputer and mainframe environments.

There are some differences, however, between MANTIS and MANTIS for the IBM mainframe, differences that result from the internal architecture of the two products and the hardware on which they run. For example, in a non-IBM environment, MANTIS is free of the constraints on screen layout imposed by the IBM 3270 display format. In addition, certain features may be available in MANTIS that is not yet available on MANTIS for the IBM mainframe.

MANTIS provides an IBM compatibility mode option to alert you to those features that are not IBM compatible. You can set IBM compatibility mode using the IBM MANTIS Option.

“IBM compatibility considerations” on page 403 describes the MANTIS features affected by IBM compatibility mode and highlights special considerations for developing applications in a non-IBM environment that will be moved to the MANTIS for the IBM mainframe environment.

Logical names

In this manual the term logical name refers to an identifier or variable that stands for another name or value.

In MANTIS for OpenVMS, the logical names used by MANTIS correspond directly to OpenVMS logical names.

In MANTIS for UNIX, logical names are implemented as environment variables. You must ensure that any shell variables you want to affect MANTIS are exported or inherited into the environment in which MANTIS is executing.

2

MANTIS language conventions

MANTIS programs are written in the MANTIS language. Before you use MANTIS, you need to know some fundamental rules of the language. This chapter is not intended to teach you how to program, but rather outlines the guidelines that MANTIS uses. Throughout this chapter, short program sequences illustrate specific aspects of the MANTIS language. You do not need to understand each statement at this point. "MANTIS programming language" on page 89 provides more details on the statements and commands that are used in the program code.

MANTIS character set

The MANTIS character set consists of:

- ♦ Alphabetic characters in uppercase (A–Z) and lowercase (a–z)
- ♦ Numeric digits (0–9)
- ♦ Special characters (defined in the following table)
- ♦ Space character (blank)

Character	Purpose
“	Double quotes enclose a text literal. (Can vary in some countries.)
‘	A single quote (apostrophe) marks a continuing line.
( )	Parentheses enclose subscripts, sub expressions, arguments or parameters.
:	A colon separates two statements on a line; also short for the SHOW command.
;	A semicolon inserts a space in a line displayed by the SHOW statement
,	A comma separates subscripts, arguments, or parameters, or inserts spaces up to the next zone in a line displayed by the SHOW statement.
.	A period designates the decimal point in a number.
_	An underline connects separate words in a symbolic name.
|	A vertical bar precedes a comment in a program line. “|” may also be entered as an exclamation point (!).
+	A plus sign adds two numeric data items or concatenates two character data items..
-	A minus sign subtracts two data items or deconcatenates two character data items. Special note: Do not use a minus sign or dash between two words in a file name (for example *file-name)* or MANTIS tries to subtract the names.
*	An asterisk multiplies one number by another.
**	A double asterisk raises one number to the power of another.
/	A slash divides one number by another
=	An equal sign tests whether one value is equal to another or assigns a value to a variable.
<	A less-than sign tests whether one value is less than another.
>	A greater-than sign tests whether one value is greater than another.
<>	A less-than sign with a greater-than sign tests whether one value is unequal to another.
>=	A greater-than sign with an equal sign tests whether one value is greater than or equal to another
<=	A less-than sign with an equal sign tests whether one value is less than or equal to another.
&	An ampersand indicates an indirect reference to a MANTIS symbolic name. See “Text expressions” on page 47 for more details.
@	An at sign directs MANTIS to obtain programming input from an external file. Available for use only in Program Design
$	A dollar sign is short for the PERFORM command.

Text considerations

This section discusses how MANTIS stores and manipulates text data.

Storing text data

MANTIS stores text data in TEXT variables. Each TEXT variable can hold a maximum of 65535 characters, although you can change this limit through the MAXSTRLEN MANTIS Option, to a maximum as low as 255 characters. Any characters in the ASCII character set can be stored in a TEXT variable. If you specify a TEXT variable as:

```
10 TEXT DATA(20)
```

MANTIS allocates storage for 20 characters of text data. You can also store text data in arrays of one or more dimensions using the TEXT statement. For example if you specify a one-dimensional array:

```
10 TEXT DATA (3,20)
```

MANTIS allocates storage as:

Data (1)	**Data (2)**	**Data (3)**

where each element can store up to 20 characters of text.

If you specify a two-dimensional array:

```
10 TEXT DATA (2,3,20)
```

MANTIS allocates storage as:

Data (1,1)	**Data (1,2)**	**Data (1,3)**
Data (2,1)	**Data (2,2)**	**Data (2,3)**

where each element can store up to 20 characters of text.

Text literals and variables

MANTIS stores text data as either literals or variables. A text literal is any string of MANTIS characters enclosed in quotes (for example, “January 7th, 1992”). Quotes within a text literal must be duplicated to distinguish them from the enclosing quotes. For example:

```
10 SHOW "THE ANSWER IS ""X""."
```

produces

```
THE ANSWER IS "X".
```

Define a TEXT variable with a TEXT statement or with a SCREEN, FILE, INTERFACE, ULTRA, ACCESS, or VIEW statement that implicitly defines TEXT variables. The default for a variable is BIG (up to sixteen significant digits), so you must define TEXT variables before using them. If you define two text variables as:

```
10 TEXT ALPHA(20),BETA(14,20)
```

MANTIS allocates 20 characters of storage for ALPHA. Since BETA is a one-dimensional array of text variables, MANTIS allocates memory for 14 elements, where each element can have a maximum of 20 characters.

A text variable or array element cannot exceed 65535 characters in length and has two length characteristics:

- A maximum length as defined in the TEXT statement
- A current length as maintained by MANTIS. The current length indicates how many characters the text variable currently contains.

If, for example, you enter:

```
10 TEXT ALICE(13,20)
```

MANTIS creates a text array with 13 elements, each with a defined maximum length of 20 characters and a current length of zero characters. If you then execute:

```
20 ALICE(6) = "TWAS BRILLIG"
```

The sixth element in the array has a current length of 12 while all the other elements still have a current length of zero.

You can use the SIZE function to obtain the current length of a text variable. For example:

```
SIZE(ALICE(6))
```

returns the value 12.

Text substrings

The value of a text variable or array element is a string of characters numbered in ascending sequence from left to right. MANTIS allows you to refer to a substring by using subscripts to specify the first and last character positions. If the second subscript is omitted, the substring includes all characters up to the end of the string. Any character positions you specify which fall outside the string are not included in the substring. For example:

```
TEXT WORDS(20)
WORDS = "LEND ME YOUR EARS"
WORDS(1,17)      refers to  LEND ME YOUR EARS
WORDS(9,12)      refers to  YOUR
WORDS(9)         refers to  YOUR EARS
WORDS(9,19)      refers to  YOUR EARS
WORDS(18)        refers to  null string
```

You can also use negative subscripts to refer to a character position relative to the end of the string:

```
WORDS(-17,-1)    refers to  LEND ME YOUR EARS
WORDS(-12,-11)   refers to  ME
WORDS(-12)       refers to  ME YOUR EARS
WORDS(-19,-11)   refers to  LEND ME
WORDS(-19)       refers to  LEND ME YOUR EARS
```

Subscripts do not have to be integers. MANTIS truncates any fraction, as if the INT function had been used.

If you refer to a substring in a text array, the position subscript(s) comes after the dimension subscript(s) that specifies the array element. For example, if ALICE(6) = “TWAS BRILLIG”:

```
ALICE(6,1,3)     refers to         TWA
ALICE(6,-6,-2)   refers to         RILLI
```

You can assign a value to a substring of a text variable or array element by using a LET statement. The substring specification is then interpreted as if the string extended to the maximum length of the text variable or array element. For example:

`WORDS(18)="."`	*sets WORDS to*	`LEND ME YOUR EARS.`
`WORDS(9)="TEN DOLLARS"`	*sets WORDS to*	`LEND ME TEN DOLLARS`

A subscript can be an arithmetic expression. If, for example, you enter:

```
10 SMALL BLOCK(3,3)
20 TEXT DANDY(6,14)
30 BLOCK(1,2) = 1
40 BLOCK(2,3) = 2
50 BLOCK(3,1) = 3
60 BLOCK(3,3) = 5
70 DANDY(6) = "DRESS WELL"
80 SHOW DANDY((BLOCK(3,1)*BLOCK(2,3)), BLOCK(1,2),
90' 4+BLOCK(3,3))
```

Lines 80 and 90 go through the following internal computational steps:

```
SHOW DANDY((3*2),1,4+5)
SHOW DANDY(6,1,9)
```

which results in:

```
DRESS WEL
```

See the last example for the SIZE statement under “SIZE” on page 339 for a routine to sum values in a one-dimensional array.

Text expressions

A text expression in MANTIS consists of one or more text operands. The following table lists the various types of text operands with an example of each type.

Operand	Example
Text Variable	NOUN, VERB
Text Array Element	ALICE(6)
Text Substring	WORDS(-8,-3)
Text Literal	“Dollars”, “50%”
Built-in Text Function	TXT(PI), DATE
Subexpression in Parentheses	(KEY+ “ PRESSED”)
Indirect Reference	&TEXT_NAME

The simplest type of text expression is an operand by itself (for example, NOUN). In expressions with more than one operand, each pair of operands is separated by an operator (for example, NOUN+VERB). You can join two text operands by using the plus (+) operator. If, for example, you enter:

```
10 TEXT ONE(12), THREE(24), TWO(12)
20 ONE = "RAIN IS "
30 TWO = "FALLING"
40 THREE = ONE+TWO
```

THREE contains:

```
RAIN IS FALLING
```

You can use the minus (-) operator to remove the characters in the second operand from the first operand. In the example above, if you change the assignment of THREE to:

```
50 THREE = TWO-"LL"
```

THREE then contains:

```
FAING
```

MANTIS removes only the first occurrence of the characters. You can find the point where the characters are removed by using the text expression as the argument of the built-in text function POINT (see "Programming fundamentals" on page 70). MANTIS returns a numeric value. If, for example, you alter statement 50 to:

```
50 A = POINT(TWO-"LL")
```

MANTIS sets A to a value of 3 because "LL" begins in the third position of TWO.

If the second term does not exist in the first term, no removal occurs, and the POINT value is zero. So you can remove multiple occurrences by repeated minus (-) operators. For example, the following will remove one to three occurrences of an "L" character:

```
TEXT STR(40)
…
STR=STR-“L”-“L”-“L”
```

The following removes one right parenthesis, one left parenthesis, and one dash(-), if any are contained in the string.

```
STR=STR-“(“-“)”-“-“
```

The following checks for at least 2 occurrences of "/" in text variable.

```
IF POINT(STR-“/”-“/”)
…| STR contains two or more “/” characters
END
```

Indirect variable reference

MANTIS symbolic names may be referred to indirectly. The indirect operator is the ampersand (&), which takes as its operand a text expression giving a symbolic name. The symbolic name is not validated and does not follow any of the rules governing MANTIS symbolic names. The following operands are supported:

- A symbolic name, for example, &NAME
- A text literal, for example, &“NAME”
- A text function, for example, &FORMAT(NUM,“NAME_###”)
- A text subexpression, for example, &(“NAME_”+TXT(NUM))
- Another level of indirection, for example, &&NAME

Note that the indirect operand cannot be an array variable. The operand must give a scalar text result. The indirect operator has a higher precedence than either array or variable substring subscripts. In other words, (&), the ampersand, binds more tightly to its right hand operand than any subscripts that may follow. The following text functions (which don’t accept parameters) are exceptions to this rule, since any subscripts have higher priority:

- DATE
- KEY
- LANGUAGE
- NULL
- PASSWORD
- TERMINAL
- TIME
- USER

For example, the following two indirect references give a different result even though the operands may appear to have the same value.

```
LANGUAGE="ENGLISH"
TEXT T:T="ENGLISH"
TEXT ENGLISH:ENGLISH="ABCDEFG"
TEXT ENGL:ENGL="12345"
SHOW &T(1,4) :| -->  "ABCD"
SHOW &LANGUAGE(1,4):| --> "12345"
```

An indirect reference may be used anywhere that a direct reference to a symbolic name is valid in expressions. Use parentheses where necessary to get around the operator precedence rules. For example, to reference through an array element:

```
&(ARRAY(1))
```

With the following exceptions, statements taking symbolic names as parameters can also take indirect references.

- ♦ Formal arguments in an ENTRY statement must be direct symbolic references.
- ♦ The symbolic name parameter in a USAGE command cannot be an indirect reference. 'USAGE &' can be used to locate all indirect references in the program.

Numeric considerations

This section discusses how MANTIS stores and manipulates numeric data. The numeric data types are INTEGER, SMALL, BIG and DECIMAL. SMALL and BIG are both floating point data types, which means they are stored with separate sign and exponent and the significant digits of the number are stored in binary form. DECIMAL numbers are similar to floating point in that they also contain separate sign and exponent data, but differ in the storage of the significant digits. DECIMAL numbers are stored as ASCII digit strings of up to 31 significant digits. The ASCII representation allows for greater decimal accuracy because the binary number system is not capable of storing some decimal values accurately.

This section discusses how MANTIS stores and manipulates numeric data. MANTIS numbers include:

- The digits 0–9
- A preceding plus or minus sign
- A period to designate the decimal point
- The letter E to indicate an exponent
- The letter D to indicate a DECIMAL exponent

The total range of numbers that can be stored using a given data type is called the domain of the data type.

Scientific notation

The letters E and D in scientific notation mean “times 10 to the power of the exponent.” The exponent must immediately follow the letter. Scientific notation is a general term for number representation using an exponent. MANTIS supports two types of scientific notation, E-notation and D-notation. E-notation denotes a floating-point number stored internally in floating-point format using the host hardware storage format. D-notation denotes a DECIMAL number stored internally in a mixture of binary and ASCII.

The number before the E/D is a decimal number containing the significant digits of the number. The number after the E/D (that is, the exponent) is an integer that specifies how many places the decimal point shifts (to the right if positive, to the left if negative). The following table gives examples of scientific notation and significant digits. Note that a large number can have a small number of significant digits and vice-versa.

Decimal notation	Scientific notation	Significant digits
123456789012	1.23456789012E11	12
.0000123456	1.23456E-05	6
-12300000000	-1.23E10	3
123456789012345678	1.23456789012345678D17	18

When MANTIS displays scientific notation it always uses the normalized form, which places the decimal point after the first significant digit. You can enter any number in scientific notation but you do not have to enter normalized numbers. You can place the decimal point anywhere you like in relation to the significant digits. For example, you can enter:

100 as 1E2	123000 as 123E3	1.7640 as 17640E-4	.0004 as 4D-4

INTEGER data type

INTEGER variables have to be declared either explicitly with the INTEGER statement or implicitly via a complex statement such as the FILE statement. INTEGER is defined as a signed 32-bit quantity with a domain of -2147483648 to 2147483647.

BIG and SMALL floating point data types

MANTIS stores floating-point data in binary floating-point format, with two precision options:

- BIG precision: holds up to sixteen significant digits
- SMALL precision: holds up to seven significant digits

Although up to sixteen significant digits are stored for a BIG number, the number of digits actually displayed depends on the display format.

BIG is the default data type for undeclared variables referenced in an executing MANTIS program.

The domain of floating point numbers is dependent on the underlying hardware. For example, MANTIS for OpenVMS accepts any number in the approximate range 0.3E-38 through 1.7E+38. MANTIS for UNIX accepts a much larger range of numbers, but stores them with less precision. A program fault occurs if a calculation results in a number outside the allowable range.

BIG variables provide for much greater accuracy in your calculations than SMALL variables. Use BIG variables unless you are concerned with the storage requirement of your program (BIG variables require approximately twice as much space as SMALL or INTEGER variables). Do not use SMALL when fractional accuracy is important as, for example, in currency calculations. DECIMAL provides even greater accuracy than BIG but at a cost of considerably slower arithmetic calculations.

Consider the following MANTIS program:

```
10 SMALL ALPHA
20 ALPHA = 987E7
```

In statement 10, ALPHA is a variable with a SMALL precision. Statement 20 assigns a value of 9870000000 to ALPHA. Since it has only three significant digits, it can be stored without loss of accuracy in the significant digits, although conversion to internal floating-point format and back may introduce a slight rounding error. In this case, MANTIS returns the value of ALPHA as 9870000128 after conversion and storage.

If, however, you specify the second line as:

```
20 ALPHA = 9876543210
```

which has nine significant digits, there is a loss of accuracy in the significant digits since MANTIS can accommodate only seven significant digits for a SMALL variable. After conversion and storage, MANTIS returns the value of ALPHA as 9876543488. Only the first seven significant digits are correct.

If you specify ALPHA as:

```
10 BIG ALPHA
```

you do not lose accuracy in the significant digits since a BIG specification holds up to sixteen significant digits. The rounding errors that may be incurred during conversion or calculation are much smaller with BIG variables than with SMALL variables. If you don't specify a data type, MANTIS defaults undefined variables to BIG.

DECIMAL data type

DECIMAL is defined as a signed floating point data type with 8-bit exponent and up to 31 significant digits. The maximum number of significant digits allowed is called the DECIMAL precision.

You must declare DECIMAL variables either explicitly with the DECIMAL statement or implicitly via a complex statement such as the FILE statement. DECIMAL precision is specified in the DECIMAL declaration or file design, and so on. The domain of a DECIMAL is influenced in a small way by its precision. For example, the domain of DECIMAL(7) is -9.999999D127 to 9.999999D127. and includes numbers as small as [+/-]9.9999999D-128.

D-notation is required when coding numeric constants if the exponent exceeds the limits of the BIG data type. For example, you cannot code 9E100 in a MANTIS for OpenVMS program because 100 is not a valid BIG exponent on OpenVMS processors. You would have to code the number as 9D100. Similarly, because BIG numbers are unreliable after about 14 significant digits, any constant of more than 14 significant digits should be encoded as a DECIMAL constant. The MANTIS Option "DECUSELEN" value (default of 14) controls the encoding to DECIMAL when the significant digits exceed the "DECUSELEN" option value. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.

Numeric conversion

When a MANTIS operator such as the addition operator (+) has operands with different numeric data types, one of the operands will always be converted to the type of the other before the operation is carried out. This is referred to as coercion, or, automatic conversion. The rule MANTIS uses to decide which operand to coerce is based on the potential magnitude of the two operands. The data type with the bigger potential magnitude is considered more important and so the operand with lesser potential magnitude will be converted. Magnitude does not equate to the number of significant digits supported but rather to the maximum integral size of the numbers that can be stored. Therefore numeric data types are ranked in the following order of importance, from lowest to highest.

- INTEGER
- SMALL
- BIG
- DECIMAL

SMALL operands are unconditionally converted to BIG in any case, even if there are two of them involved in an operation, which means that coercion can never be to SMALL, but will always be to either BIG or DECIMAL.

When a MANTIS program executes, numeric constants are given the data type of BIG regardless of whether they are integers (whole numbers) or not. The only exception is that DECIMAL storage format is used if either of the following conditions is met:

- The letter D is used to specify an exponent
- There are more than 14 significant digits specified (this excludes any exponent digits) than contained in the “DECUSELEN” option setting (default of 14).

This is important because the data type of a numeric constant may influence the type of arithmetic used in calculations. For example:

```
 10 INTEGER I
    .
    .
    .
100 I=I+1
```

At line 100, the addition is done using floating point arithmetic because the numeric constant, 1, is a BIG value, and therefore requires that the value of the variable, I, be coerced from INTEGER to BIG. The BIG result of I+1 then has to be converted back to INTEGER for assignment to I. For maximum efficiency, ensure that all operands in arithmetic expressions are of the same type.

How MANTIS displays numeric data

MANTIS uses several methods to display numbers with many significant digits. The CONVERSE statement always tries to display numbers in fixed notation and will do so as long as the display field size can accommodate the number of significant digits, otherwise scientific notation (E or D-notation) will be used and as much precision as necessary is discarded in order to fit the scientific representation into the field.

The SHOW statement builds up a line of output based on 13-character tab zones used in conjunction with the COMMA delimiter. Therefore numeric display fields are restricted to 12 characters, as if the numeric edit mask were "############." As with CONVERSE, scientific notation is considered less desirable and is not used unless the number cannot be displayed in the 12-character field. You can SHOW greater precision by using the FORMAT function to supply your own conversion mask.

In the 12-character field of the SHOW statement output, scientific notation is used to display a number if any of the following conditions is met.

- The number is greater than or equal to 1E12
- The number is greater than -1E-4 and less than 1E-4,   that is, between -0.0001 and 0.0001.
- The number is less than or equal to -1E11

When listing a MANTIS program, scientific notation is used to display a number if either of the following conditions is met.

- A floating point constant would take more than 16 digits to display in fixed-point format.
- A DECIMAL constant would take more than 31 digits to display in fixed-point format.

The purpose of these rules is to use scientific notation in program listings only if there are more digits in a number than the precision allowed for the data type of the number, which is 31 for DECIMAL and 16 for floating point.

Numeric arrays

A numeric array is an ordered set of numeric values. Each value is called an array element. Numeric arrays can have from 1 to 255 dimensions. Each dimension can be in the range 1–65535.

Define numeric arrays with INTEGER, BIG, SMALL and DECIMAL statements, or via a levels specification on an ACCESS, FILE, INTERFACE, ULTRA, or VIEW statement. Numeric arrays are automatically created on execution of an ACCESS, FILE, INTERFACE, SCREEN, or ULTRA statement when the definition contains repeating fields or elements.

MANTIS allocates storage to hold array elements as follows. If, for example, you specify a one-dimensional array:

```
10 BIG MAN(5)
```

MANTIS allocates storage as:

MAN(1)	MAN(2)	MAN(3)	MAN(4)	MAN(5)

If you specify a two-dimensional array:

```
10 BIG SAM(3,3)
```

MANTIS allocates storage as:

Sam (1,1)	**Sam (1,2)**	**Sam (1,3)**
Sam (2,1)	**Sam (2,2)**	**Sam (2,3)**
Sam (3,1)	**Sam (3,2)**	**Sam (3,3)**

To access one element of an array, you must specify its position within the array by subscripting the array variable. A subscript can be either a number or an arithmetic expression. For example, if you have an array with 17 elements:

```
10 BIG STAR(17)
```

You can access the first element by entering STAR(1); you can access the eleventh element in any of the following ways:

```
STAR(11)  or  K=11    or  STAR(6+5)      or     STAR(1)=9
STAR(K)                         K=2
STAR(STAR(1)+K)
```

The value you use to subscript an array must be between one and the maximum dimensions given for that array. If the value is outside this range, MANTIS returns an error message. If, for example, you specify:

```
10 SMALL NEUTRON(7,3)
20 NEUTRON(6,4) = 2
```

MANTIS displays an error message because the second dimension (4) in statement 20 exceeds the maximum (3) defined in statement 10.

Arithmetic expressions

An arithmetic expression in MANTIS consists of one or more numeric operands. The following table lists the various types of numeric operands with an example of each type.

Operand	Example
Numeric Variable	ALPHA, BETA, GAMMA
Numeric Array Element	A(5), C(2), X(Y+Z)
Numeric Constant	1.43, 0.07, 14E-7
Built-in Numeric Function	SIN(Y), NOT(Z)
Built-in Numeric Constant	PI, E, FALSE (0)
Subexpression in Parentheses	(4+SIN(Y))
Indirect Reference	&BIG_NAME

The simplest type of expression is an operand by itself (for example, ALPHA). In expressions with more than one operand, each pair of operands is separated by an operator (for example, ALPHA+ BETA-GAMMA). An operator is a symbol for an arithmetic operation (for example, + is the symbol for addition). The following table lists the MANTIS arithmetic operators with a brief description of their operations and an example of each one.

Symbol	Operation	Example
**	Raise A to the power B.	A**B
*	Multiply A by B.	A*B
/	Divide A by B. Note that if B is equal to zero, MANTIS sets the result to zero to avoid the error condition when dividing by zero.	A/B
+	Add A to B.	A+B
-	Subtract B from A.	A-B
=	If A and B have the same value, the expression evaluates to TRUE, that is (1); otherwise, it evaluates to FALSE, that is, (0). Also indicates the assignment of a value to a variable (for example, LET X=Y).	A=B

Symbol	Operation	Example
<	If the value of A is less than the Value of B, the expression evaluates to TRUE (1), otherwise, it evaluates to FALSE (0)	A < B
>	If the value of A is greater than the value of B, the expression evaluates to TRUE (1); otherwise, it evaluates to FALSE (0).	A > B
<=	If the value of A is less than or equal to the value of B, the expression evaluates to TRUE (1); otherwise, it evaluates to FALSE (0).	A <= B
>=	If the value of A is greater than or equal to the value of B, the expression evaluates to TRUE (1); otherwise, it evaluates to FALSE (0).	A >= B
<>	If the value of A does not equal the value of B, the expression evaluates to TRUE (1); otherwise, it evaluates to FALSE (0).	A <> B
AND	If both operands are TRUE (that is, nonzero) the expression evaluates to TRUE (1); otherwise, it evaluates to FALSE (0).	A AND B (A=B) AND (C=D)
OR	If either operand (or both) is TRUE (nonzero), the expression evaluates to TRUE (1); otherwise, it evaluates to FALSE (0).	(A=B) OR (C=D)

Note that if you have the expression A/B, and B is equal to zero (normally an infinitely large number), MANTIS sets the value of the expression to zero to avoid error conditions.

The first operand in an expression can be preceded by a + or - operator which is then called a unary operator (for example, -ALPHA+BETA). The unary operator - changes the sign of the first operand. The unary operator + has no effect. To change the sign of an operand other than the first, you must enclose the operand in parentheses so that it becomes the first operand of a sub-expression (for example, ALPHA/(-GAMMA)).

MANTIS evaluates an arithmetic expression by applying the operators to the operands in the following order:

1. UNARY + -
2. **
3. *, /
4. +, -

MANTIS evaluates a subexpression enclosed in parentheses before the expression in which it is an operand. For example, if you enter:

```
YEAR+DAY/(6*(MONTH-LAG))
```

MANTIS subtracts LAG from MONTH, multiplies the result by 6, divides DAY by the product and adds YEAR to the result of the division.

When two operators have equal priority (for example, + -), MANTIS evaluates the expression from left to right.

You can use an ampersand (&) to make an indirect reference to a numeric variable whose symbolic name is stored in a text variable. For example:

```
TEXT BIG_NAME
BIG ONES,TWOS
ONES=111111
TWOS=222222
BIG_NAME="ONES"
SHOW &BIG_NAME
111111
BIG_NAME="TWOS"
SHOW &BIG_NAME
222222
```

An indirect reference can appear as an operand anywhere in an expression. If the symbolic name referred to indirectly is not defined, MANTIS automatically defines it as a BIG variable.

MANTIS keywords

MANTIS uses keywords to identify commands, statements, built-in functions, and constants. Keywords must be entered exactly as they appear and cannot be used for any other purpose. The following table shows MANTIS keywords.

NOTE

The table is in alphabetical order for two pages. Each column extends beyond the first page to the second, and then continues back on page one, i.e., the “end” keyword is the last entry on the next page of the table, but the “enqueue” keyword is the first entry of the second column on this page of the table.

$LOGICAL	ENQUEUE	MIXMODE	SGN
$OPTION	ENTRY	MIXT	SHOW
$SYMBOL	EQUAL	MODIFIED	SIN
ABS	ERASE	NEW	SIZE
ACCESS	EXEC_SQL	NEXT	SLICE
AFTER	EXECUTE	NOT	SLOT
ALL	EXIT	NULL	SMALL
ALTER	EXP	NUMERIC	SPECTRA
AND	FALSE	OBTAIN	SQLBIND
ASI	FILE	OFF	SQLCA
AT	FIRST	ON	SQLDA
ATN	FOR	OR	SQR
ATTRIBUTE	FORMAT	ORD	STATS
BEFORE	FSI	OUTPAD	STOP
BIG	GET	PAD	SUBMIT
BIND	GO	PASSWORD	TAN
BREAK	HEAD	PERFORM	TERMINAL
BY	HELP	PERM	TERMSIZE
CALL	IF	PI	TEXT
CHAIN	INSERT	POINT	TIME
CHANGE	INT	POSITION	TO
CHR	INTEGER	PREFIX	TOTAL
CLEAR	INTERFACE	PRINTER	TRAP

CMS	INTERNAL	PRIOR	TRUE
COMMIT	KANJI	PROGFREE	TXT
CONVERSE	KEY	PROGRAM	ULTRA
COPY	KEYF	PROMPT	UNPAD
COS	KEYM	PURGE	UNTIL
CURSOR	KEYR	QUIT	UP
DATAFREE	KEYS	RELEASE	UPDATE
DATE	LANGUAGE	REPLACE	UPPERCASE
DBCS	LAST	RESET	USAGE
DBPAGE	LET	RETURN	USER
DECIMAL	LEVEL	RND	USERWORDS
DELETE	LIST	ROUNDED	VALUE
DEQUEUE	LOAD	ROUNDING	VIA
DISPLAY	LOG	RUN	VIEW
DO	LOWERCASE	SAME	VSI
DOLEVEL	MARK	SAVE	WAIT
DOWN	MASK	SCREEN	WHEN
E	MEMORY	SCROLL	WHILE
EDIT	MIXD	SEED	WINDOW
EDITRCV	MIXED	SELECT	ZERO
ELSE	MIXED	SEQUENCE	
END	MIXM	SET	

MANTIS symbolic names

MANTIS uses symbolic names to represent data processed by a MANTIS program. Symbolic names may stand for either numeric or text data. MANTIS allows a maximum of 65536 symbolic names for a single program, including names defined implicitly by SCREEN, INTERFACE, FILE, ACCESS, ULTRA, and VIEW statements.

A symbolic name:

- Must begin with an alphabetic character.
- Is limited to 80 characters in length.
- May contain alphabetic and numeric characters and the underline (_). No other special characters are allowed. If you enter a name using lowercase characters, MANTIS converts them to uppercase when you press RETURN. (for example, the following variables are equivalent: customer_name, Customer_Name, CUSTOMER_NAME).
- Must not be a keyword as listed in "MANTIS keywords" on page 63. A symbolic name may contain a keyword (for example, LASTONE) but may not be a keyword in its entirety (for example, LAST).
- Can be any size that fits on a line. However, if the field is used in a design entity, (for example, screens, interfaces, or files) it is limited to 16 or 30 characters.
- Must be unique. MANTIS ignores any attempts to redefine a symbolic name that already exists. If MANTIS encounters a definition for a symbolic name for the second time (a duplicate definition), it does not create a second working area for that name. However, with variables defined within complex statements such as ACCESS, MANTIS checks the current and new data type for consistency.
- MANTIS automatically converts all symbolic names and reserved words to uppercase in your program.
- MANTIS reserves certain words for command names, built-in functions, and other features of the language.
- If you use the "PREFIX" option on complex statement, the variable name will be appended to the complex entity name to form a symbolic name (for example FILENAME_FIELDNAME).

Relational expressions

A relational expression is an expression that evaluates to a value of TRUE (1) or FALSE (0), depending on the relationship between its operands. MANTIS has no special data type for the logic values TRUE (1) and FALSE (0) that are treated as numeric values. All nonzero values represent TRUE (1); zero represents FALSE (0).

MANTIS relational operators compare two numeric operands or two text operands and produce a result of TRUE (1) or FALSE (0). The MANTIS relational operators are shown in the following table.

Symbol	Operation	Example
=	If A equals B, the value of A=B is TRUE (1); otherwise, the value is FALSE (0).	A=B
<	If A is less than B, the value of A<B is TRUE (1); otherwise, the value is FALSE (0).	A<B
>	If A is greater than B, the value of A>B is TRUE (1); otherwise, the value is FALSE (0)	A>B
<>	If A does not equal B, the value of A<>B is TRUE (1); otherwise, the value is FALSE (0).	A<>B
>=	If A is greater than or equal to B, the value of A>=B is TRUE (1); otherwise, the value is FALSE (0).	A>=B
<=	If A is less than or equal to B, the value of A<=B is TRUE (1); otherwise, the value is FALSE (0)	A<=B

When comparing text operands, MANTIS uses either ALPHANUMERIC or LENGTH based comparison, depending upon the setting of the String Comparison option in the program context. When a MANTIS program context is created, the default String Comparison option is inherited from the STRCOMPARE MANTIS Option.

ALPHANUMERIC comparison of text operands compares them up to the length of the shorter text operand and compares their lengths only if they are found to be equal up to the shorter length. MANTIS compares from left to right, character-by-character, using the standard ASCII collating sequence. For example, the value of “MEN” > “MANTIS” is TRUE (1) because “E” is higher in collating sequence than “A”; however, “MAN” > “MANTIS” is FALSE (0) because the operands are equal up to three characters, but the SIZE of MAN is less than that of MANTIS.

LENGTH comparison of text operands compares their lengths first and compares their values only if the lengths are equal. For example, the value of “MANHATTAN” > “NEW YORK” is TRUE (1) because the SIZE of MANHATTAN is nine characters and that of NEW YORK is eight characters. Using ALPHANUMERIC comparison, the value of “MANHATTAN” > “NEW YORK” is FALSE (0) because “M” is lower in collating sequence than “N.”

Once a program is created, you cannot change its String Comparison option except in the Program Design Facility library option.

Symbol	**Operation**	**Example**
AND	If A is TRUE (1) and B is TRUE (1), the value of A AND B is TRUE (1); otherwise, the value is FALSE (0)	A AND B
OR	If A is TRUE (1) or B is TRUE (1), the value of A OR B is TRUE (1); otherwise, the value is FALSE (0).	A OR B

You can mix operators of all types (that is, arithmetic, text, relational, and logical) in a relational expression (for example, A<B+C OR A>D*E). MANTIS evaluates the operators in the following order:

1. UNARY + -
2. **
3. * /
4. + -
5. =, >, <, >=, <=, <>
6. AND
7. OR

When the operators have equal priority and parentheses exist, MANTIS evaluates the expression from left to right.

The equivalence of numeric and logical values allows an arithmetic expression to be used in place of a relational expression and vice versa, although the result may not be meaningful. While accepting any nonzero value as TRUE (1), MANTIS always evaluates a TRUE result to 1. FALSE always evaluates to 0.

Note that both terms of AND and OR are evaluated, in spite of the value of the first term. For example, in the expression IF A AND B, both A and B are evaluated even if A is FALSE (0). Therefore, IF I > 0 AND X(I)=3 will fail if I <= 0 since X(I) will be evaluated and the value of I is not a valid array subscript.

You can use logical expressions to simplify code that otherwise needs IF or WHEN constructions. For example, to determine payment terms, you can replace:

```
IF CREDIT_RATING="A"
.TERMS=30
ELSE
.IF CREDIT_RATING="B"
..TERMS=10
.ELSE
..TERMS=0
.END
END
```

with:

```
10 TERMS=((CREDIT_RATING="B")*10)+((CREDIT_RATING="A")*30)
```

Programming fundamentals

You can use the MANTIS Program Design Facility and/or an external full-screen editor to enter and maintain your MANTIS programs. For more information about the Program Design Facility and the EDIT command, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300. This section provides an overview of MANTIS programming fundamentals.

In MANTIS programming mode, you can enter two types of input:

- **Statements:** The smallest unit of execution in MANTIS and consist of a reserved word and, if necessary, one or more operands. A statement can be part of a program line and require a run-mode action (such as FILE and SCREEN) or can be issued for immediate execution in programming mode. "MANTIS programming language" on page 89 provides information on MANTIS statements.

- **Commands**: Special statements that can only be issued for immediate-mode execution from the MANTIS Program Design Facility (refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information). Commands cannot be included as part of a MANTIS program.

Some of the MANTIS reserved words can be used as both statements and commands (such as EXIT). Most MANTIS statements can be entered without a line number and executed immediately (such as SHOW and LET). You don't need to remember which reserved words are statements and which are commands. MANTIS returns an error message if you use them incorrectly. You can run a partial or completed program at any point by entering the RUN command. Your program executes from the first to the last coded statement. MANTIS interprets programs as they run; with no compiler and no object code. MANTIS first checks the logical structure of your program to ensure matched statements, such as IF-END, ENTRY-EXIT.

- You can save a program that is not logically complete (such as missing an END statement), but the program must be logically balanced to RUN it. Refer to the SAVE command in *AD/Advantage MANTIS Facilities OS/390, VSE/ESA*, P39-5001.

MANTIS “interprets” each statement of your program as it runs; there is no compiler and no object code. MANTIS terminates a program when:

- MANTIS executes a STOP statement.
- MANTIS encounters no more statements to process (physical end of the program)
- MANTIS encounters an error.
- You press CTRL-C (unless this feature is turned off by the NOCRTLC MANTIS Option).
- You enter KILL in the Reply Field (or press the GOLD/K key) in response to program input.

Statements

A program statement consists of a line number (1 through 64000), a MANTIS keyword, and, in many cases, one or more operands. You can use spaces freely to make statements easier to read. MANTIS ignores spaces except when they appear in text literals or perform a delimiting function. For example, the following two statements have the same meaning to MANTIS:

```
120 SHOW  ALPHA,  BETA,   - 1500+ 7
130 SHOW  ALPHA,BETA,   -1500 + 7
```

When you use the LIST command, MANTIS removes all unnecessary blanks and inserts necessary blanks as follows:

```
LIST
   120 SHOW ALPHA,BETA,-1500+7
   130 SHOW ALPHA,BETA,-1500+7
```

In addition, when you list a program, periods (.) may appear between the statement number and the keyword. MANTIS inserts these periods automatically to show your program’s nesting hierarchy. You do not need to supply periods when you enter statements.

```
LIST
   120 ....SHOW ALPHA,BETA,-1500+7
   130 ....SHOW ALPHA,BETA,-1500+7
```

Statements can contain as many characters as will fit on the line. If you need more characters, you can continue the line by entering an apostrophe (‘) immediately after the next line number:

```
15 SHOW "THE RELATIVE HUMIDITY IS"; HMD; "AND THE TEMPERATURE IS";
16 TEMP
```

Do *not* separate a symbolic name (such as TEMP) or a number (such as 12003). Both of the following continuation attempts are *incorrect*. In the first example, instead of showing the value of TEMP, two values would be shown (TE and MP). If there are not any fields TE and MP, then fields will be defined by default with these names as BIG variables. In the second example, 12003 is treated as two separate numbers (12 and 003).

```
15 SHOW "THE RELATIVE HUMIDITY IS"; HMD; "AND THE TEMPERATURE IS"; TE
16' MP
17 SHOW "THE RELATIVE HUMIDITY IS"; HMD; "AND THE POLLEN COUNT IS";12
18' 003
```

If you need to continue a long literal (that is, text enclosed in quotes), close the literal on the first line (closing quotation mark (”)) and reopen it (opening quotation mark (“)) on the continuation line:

```
30 SHOW "THIS IS THE WEATHER REPORT FOR ALL REGIONS SOUTH AND "
35' "SOUTHWEST OF THE VALLEY"
```

You can include two or more statements on the same line by separating them with a colon (:).

```
20 CLEAR : HEAD "THIS APPEARS AT THE TOP OF THE SCREEN"
```

You cannot, however, include a statement with a logic keyword on a line with another statement. You can, however, include any text after a BREAK, ELSE, NEXT, END, or RETURN. The MANTIS logic keywords are:

BREAK	FOR
CHAIN	IF
DO	NEXT
ELSE	RETURN
END	UNTIL
ENTRY	WHEN
EXIT	WHILE

Commands

MANTIS executes a command immediately. A MANTIS command consists of a keyword and, in many cases, an operand. You use commands to RUN a program, ALTER a line, or LIST a program. You can use some statements as commands by not entering a line number. The following example illustrates SHOW as a statement and a command.

```
10 ENTRY COMPOUND
20 .SHOW"WHAT IS THE CAPITAL AMOUNT?"        <-- statement
30 .OBTAIN INVESTMENT
40 EXIT
RUN
WHAT IS THE CAPITAL AMOUNT?
1400
SHOW INVESTMENT+650                          <-- command
2050
```

MANTIS program design commands is documented in *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300. These commands are:

ALTER	GO	REPLACE
BIND	HELP	RUN
CHANGE	KILL	SAVE
CLEAR-BREAK	LIST	SEQUENCE
COPY	LOAD	SET BREAK
DOWN	NEW	SHOW
EDIT/EDITRCV	PURGE	UP
ERASE	QUIT	USAGE

Comments

You can insert a comment anywhere in your program by entering a statement number, a vertical bar (**|**), and your comment

```
15 | THIS IS AN EXAMPLE
LIST
   10 ENTRY COMPOUND
   15 .| THIS IS AN EXAMPLE
   20 .SHOW "WHAT IS THE CAPITAL AMOUNT?"
   30 .OBTAIN INVESTMENT
   40 EXIT
```

You can also follow a statement with a comment on the same line by entering a colon, a vertical bar (**|**), and your comment:

```
10 ENTRY COMPOUND:| THIS IS AN EXAMPLE
```

You can use all characters in the MANTIS character set in your comments. MANTIS ignores the rest of the line when it encounters a comment during program execution. A comment line cannot be continued on the next line by using the apostrophe.

Built-in functions

Many numeric and text functions are built into MANTIS. The following table describes the MANTIS built-in functions using the following convention to indicate argument types.

a = an arithmetic expression

f = a field name defined by a SCREEN design

m = a text edit mask for formatting numeric data

s = symbolic name for a SCREEN

t = a text expression

v = a symbolic VIEW name

x = a text or arithmetic expression, variable or array

Functions that do not have any arguments are sometimes referred to as built-in constants or built-in variables. You can use subscripts to obtain a substring of a built-in text variable. All of the functions in the following table appear with a full description in “MANTIS programming language” on page 89.

Online help is available for MANTIS built-in functions. Simply enter HELP and the function name. In cases where the function has the same keyword as a MANTIS statement (for example, ATTRIBUTE), identify the function by entering the keyword, followed by a set of parentheses (for example, HELP ATTRIBUTE()).

Function	Description	Input	Output	Type of function
ABS(*a*)	Returns the absolute value of "*a*." MANTIS evaluates "*a*" and changes the sign of the result if it is negative.	Numeric	Numeric	Mathematical
access-name	Returns a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on an external file.	*	Text	Status
ASI(*v*,*f*)	Indicates the status of a field in an RDM user view.	Field-name	Text	File Access
ATN(*a*)	Returns the angle in radians whose tangent is "*a*."	Numeric	Numeric	Mathematical
ATTRIBUTE	Returns the attributes of a field or device or the physical coordinates of the cursor on the screen.	Map or Field-name / Reserved word	Text	System
CHR(*a*,...)	Returns a text value consisting of the character(s) with ASCII code(s) as specified.	Numeric	Text	String
COS(*a*)	Returns the cosine of "*a*," where "*a*" is in radians.	Numeric	Numeric	Mathematical

Function	**Description**	**Input**	**Output**	**Type of function**
CURSOR	Determines the location of the cursor during the last terminal I/O.	Map or Field-name or Reserved word	Numeric or Text	System
DATAFREE	Returns the number of free bytes in the data work area.	*	Numeric	System
DATE	Returns the text value of the current date.	*	Text	System
DOLEVEL	Returns your current level in an external subroutine.	*	Numeric	System
E	Returns the value of natural e (2.71828182845905).	*	Numeric	Mathematical
EXP(*a*)	Returns the value of natural e to the power of "*a*."	Numeric	Numeric	Mathematical
FALSE	Returns the value FALSE (0).	*	Numeric	Boolean
file-name	Returns a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on a MANTIS file.	*	Text	Status
FORMAT(*a*,*m*)	Returns a text-string conversion of a numeric expression according to a supplied edit mask.	Numeric	Text	System

Function	**Description**	**Input**	**Output**	**Type of function**
FSI(*v*)	Indicates the status of a GET, DELETE, INSERT, or UPDATE operation on an RDM user view, external file view, or MANTIS file.	File-name	Text	File Access
INT(*a*)	Returns the integer value of "*a*," truncating fractions toward zero.	Numeric	Numeric	Mathematical
interface-name	Returns the text value stored in the status variable by the interface program	*	Text	Status
KEY	Returns a text value identifying your response to the most recent CONVERSE, OBTAIN, PROMPT, or WAIT statement.	*	Text	System
LANGUAGE	Returns the default language of the user who is currently signed-on	*	Text	System
LOG(*a*)	Returns the natural logarithm of "*a*."	Numeric	Numeric	Mathematical
$LOGICAL	Allows access to logical names from a MANTIS program.	Text	Text	System
LOWERCASE(*t*)	Returns the text value of "*t*" with any uppercase letters changed to lowercase.	Text	Text	String
MODIFIED(*s*,*f*)	Tests whether a specific field or number of fields within a map definition changed during the last terminal I/O.	Map or Field-name	Numeric	System

Function	Description	Input	Output	Type of function
NOT(*a*)	Returns TRUE(1) if "*a*" evaluates to FALSE(0);otherwise, returns FALSE.	Numeric	Numeric	Boolean
NULL	Returns a zero length text value.	*	Text	String
NUMERIC(*t*)	Determines if a text expression contains a valid number	Text	Numeric	String
$OPTION(*t*)	Returns the value of a MANTIS option.	Text	Numeric or Text	System
PASSWORD	Returns a text value containing the password entered during MANTIS sign-on.	*	Text	System
PI	Returns the value of Pi (3.14159265358979).	*	Numeric	Mathematical
POINT(*t*+*t*)	Returns a number representing the position where a string addition or subtraction would occur if you executed it.	Text	Numeric	String
PRINTER	Returns a text value containing the current printer assignment.	*	Text	System
PROGFREE	Returns the number of free bytes in the program work area.	*	Numeric	System
RND(*a*)	Returns a random real number in the range zero to "*a*," but excluding zero and "*a*."	Numeric	Numeric	Mathematical

Function	Description	Input	Output	Type of function
screen-name	Returns a text value identifying your response to the most recent CONVERSE statement for the screen.	*	Text	Status
SGN(*a*)	Returns -1 if *a*<0; 0 if *a*=0; +1 if *a*>0.	Numeric	Numeric	Mathematical
SIN(*a*)	Returns the sine of "*a*" where "*a*" is in radians.	Numeric	Numeric	Mathematical
SIZE	Return the size and dimensions of an expression, variable, or array.	Text or Variable-name	Numeric	String
SQR(*a*)	Returns the square root of "*a*."	Numeric	Numeric	Mathematical
$SYMBOL	Returns the text value assigned to a symbol.	Text	Text	System
TAN(*a*)	Returns the tangent of "*a*" where "*a*" is in radians.	Numeric	Numeric	Mathematical
TERMINAL	Returns a text character string of 1–8 characters containing the terminal ID.	*	Text	System
TERMSIZE	Returns a text value giving the terminal size in rows and columns.	*	Text	System
TIME	Returns a text value expressing the current system time.	*	Text	System
TRUE	Returns the value +1.	*	Numeric	Boolean
TXT(*a*)	Returns the text value of "*a*," in numeric display format.	Numeric	Text	String

Function	Description	Input	Output	Type of function
ultra-name	Returns a text value indicating the status of the most recent GET, INSERT, UPDATE, or DELETE operation performed on an ULTRA file.	*	Text	Status
UPPERCASE(*t*)	Returns the text value of "*t*" with any lowercase letters changed to uppercase.	Text	Text	String
USER	Returns a 1–16 text value containing the user name entered during MANTIS sign-on.	*	Text	System
USERWORDS	Returns the number of MANTIS symbolic names currently in use.	*	Numeric	System
VALUE(*t*)	Returns the numeric value of the text expression "*t*."	Text	Numeric	String
view-name	Returns a text value indicating the status of the most recent GET, INSERT, UPDATE, or DELETE operation performed on an RDM user view.	*	Text	Status
VSI(*v*)	Indicates the highest status of an RDM user view field.	File-name	Text	File Access or System
ZERO	Returns the value zero.	*	Numeric	Mathematical

*This built-in function has no arguments.

Built-in functions by type of function

The following tables are a reorganization of the previous table. While the table in “Built-in functions” on page 75 is organized alphabetically, this is organized by the type of function, either Boolean, File Access, Mathematical, Status, String, or System. Use this table when you want to perform a certain task, but are unsure of the particular function you will need. You can scan the particular category to see other related functions as well.

Boolean functions

Function	Description	Input	Output
FALSE	Returns the value FALSE (0).	*	Numeric
NOT(*a*)	Returns TRUE (1) if “*a*” evaluates to FALSE (0); otherwise, returns FALSE.	Numeric	Numeric
TRUE	Returns the value +1.	*	Numeric

*This built-in function has no arguments.

File access functions

Function	Description	Input	Output
ASI(*v*,*f*)	Indicates the status of a field in an RDM user view.	Field-name	Text
FSI(*v*)	Indicates the status of a GET, DELETE, INSERT, or UPDATE operation on an RDM user view, external file view, or MANTIS file.	File-name	Text
VSI(*v*)	Indicates the highest status of a field in an RDM user view.	File-name	Text

Mathematical functions

Function	Description	Input	Output
ABS(*a*)	Returns the absolute value of "*a*." MANTIS evaluates "*a*" and changes the sign of the result if it is negative.	Numeric	Numeric
ATN(*a*)	Returns the angle in radians whose tangent is "*a*."	Numeric	Numeric
COS(*a*)	Returns the cosine of "*a*" where "*a*" is in radians.	Numeric	Numeric
E	Returns the value of natural e (2.71828182845905).	*	Numeric
EXP(*a*)	Returns the value of natural e to the power of "*a*."	Numeric	Numeric
INT(*a*)	Returns the integer value of "*a*," truncating fractions toward zero.	Numeric	Numeric
LOG(*a*)	Returns the natural logarithm of "*a*."	Numeric	Numeric
PI	Returns the value of Pi (3.14159265358979).	*	Numeric
RND(*a*)	Returns a random real number in the range zero to "*a*," but excluding zero and "*a*."	Numeric	Numeric
SGN(*a*)	Returns –1 if *a* <0; 0 if *a*=0; +1 if *a*>0.	Numeric	Numeric
SIN(*a*)	Returns the sine of "*a*" where "*a*" is in radians.	Numeric	Numeric
SQR(*a*)	Returns the square root of "*a*."	Numeric	Numeric
TAN(*a*)	Returns the tangent of "*a*" where "*a*" is in radians.	Numeric	Numeric
ZERO	Returns the value zero.	*	Numeric

*This built-in function has no arguments.

Status functions

Function	Description	Input	Output
access-name	Returns a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on an external file.	*	Text
file-name	Returns a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on a MANTIS file.	*	Text
interface-name	Returns the text value stored in the status variable by the interface program	*	Text
screen-name	Returns a text value identifying your response to the most recent CONVERSE statement for the screen.	*	Text
ultra-name	Returns a text value indicating the status of the most recent GET, INSERT, UPDATE, or DELETE operation performed on an ULTRA file.	*	Text
view-name	Returns a text value indicating the status of the most recent GET, INSERT, UPDATE, or DELETE operation performed on an RDM user view.	*	Text

*This built-in function has no arguments.

String functions

Function	Description	Input	Output
CHR(*a*,…)	Returns a text value consisting of the character(s) with ASCII code(s) as specified.	Numeric	Text
LOWERCASE(*t*)	Returns the text value of "*t*" with any uppercase letters changed to lowercase.	Text	Text
NULL	Returns a zero length text value.	*	Text
$OPTION(*t*)	Returns the value of a MANTIS option.	Text	Numeric/ Text
POINT(*t*+*t*)	Returns a number representing the position where a string addition or subtraction would occur if you executed it.	Text	Numeric
SIZE	Returns the size and dimensions of an expression, variable, or array.	Text/ Variable-name	Numeric
$SYMBOL	Returns the text value assigned to a symbol.	Text	Text
TXT(*a*)	Returns the text value of "*a*," in numeric display format.	Numeric	Text
UPPERCASE(*t*)	Returns the text value of "*t*" with lowercase letters changed to uppercase.	Text	Text
VALUE(*t*)	Returns the numeric value of the text expression "*t*."	Text	Numeric

*This built-in function has no arguments.

System functions

Function	Description	Input	Output
ATTRIBUTE	Returns the attributes of a field or device or the physical coordinates of the cursor on the screen.	Map/ Field-name/ Reserved word	Text
CURSOR	Determines the location of the cursor during the last terminal I/O.	Map/ Fieldname/ Reserved word	Numeric / Text
DATAFREE	Returns the number of free bytes in the data work area.	*	Numeric
DATE	Returns the text value of the current date.	*	Text
DOLEVEL	Returns your current level in an external subroutine.	*	Numeric
FORMAT(*a*,*m*)	Returns a text-string conversion of a numeric expression according to a supplied edit mask.	Numeric	Text
KEY	Returns text value identifying your response to the most recent CONVERSE, OBTAIN, PROMPT, or WAIT statement.	*	Text
LANGUAGE	Returns the default language of the user who is currently signed-on.	*	Text
$LOGICAL	Allows access to logical names from a MANTIS program.	Text	Text
MODIFIED(*s*,*f*)	Tests whether a specific field or number of fields within a map definition changed during the last terminal I/O.	Map/ Fieldname	Numeric
PASSWORD	Returns a text value containing the password entered during MANTIS sign-on.	*	Text
PRINTER	Returns a text value containing the current printer assignment.	*	Text

Function	Description	Input	Output
PROGFREE	Returns the number of free bytes in the program work area.	*	Numeric
TERMINAL	Returns a text character string of 1–8 characters containing the terminal ID.	*	Text
TERMSIZE	Returns a text value giving the terminal size in rows and columns.	*	Text
TIME	Returns a text value expressing the current system time.	*	Text
USER	Returns a 1–16 text value containing the user name entered during MANTIS sign-on.	*	Text
USERWORDS	Returns the number of MANTIS symbolic names currently in use.	*	Numeric

*This built-in function has no arguments.

Summary

“MANTIS programming language” on page 89 provides an alphabetical listing and description of the statements, and built-in functions you can use to build your MANTIS programs. Commands for designing and editing your MANTIS programs are listed and described in *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.

3

MANTIS programming language

In addition to describing each statement and built-in function, this chapter provides the following information under the following headings:

Description	A description of the parameter
Default	A default value, if any applies to the parameter
Format	The required format of the parameter
Options	Choices available for the parameter
Example	Code example using the statement or function

Considerations

- Any special limitations, considerations, and guidelines for specific parameters

General considerations

- Any special limitations, considerations, and guidelines for the overall statement or function.

NOTE

Some statements and functions can be used with MANTIS, but are not supported by MANTIS for the IBM mainframe. These statements and functions are noted as such in their first consideration. "IBM compatibility considerations" on page 403 discusses compatibility between the two environments in more detail. "Programming techniques" on page 415 discusses programming techniques.

Summary of MANTIS statements and functions

The following table lists and describes the statements and functions in this chapter by type (some are more than one type). The conventions shown below are used to indicate argument types. Also included is the mode (either “Run or Immediate” or “Run”). Some entries in the table, such as ATTRIBUTE, are both statements and functions.

a = an arithmetic expression

f = a field name defined by a SCREEN design

m = a text edit mask for formatting numeric data

s = symbolic name for a SCREEN

t = a text expression

v = a symbolic VIEW name

Name	**Types**	**Description**	**Mode**
ABS(*a*)	Function	Returns the absolute value of “*a*.” MANTIS evaluates “*a*” and changes the sign of the result if it is negative.	Run or Immediate
ACCESS	Statement	Identifies an external file that your program will access.	Run or Immediate
access-name	Function	Returns a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on an external file.	Run or Immediate
ASI (statement)	Statement	Sets a null value for a field.	Run or Immediate
ASI (function)	Function	Indicates the status of a field in an RDM user view.	Run or Immediate
ATN(*a*)	Function	Returns the angle in radians whose tangent is “*a*.”	Run or Immediate

Name	Types	Description	Mode
ATTRIBUTE (statement)	Function, Statement	Alters (or returns) the attributes of a field or device or the physical coordinates of the cursor on the screen.	Run or Immediate
BIG	Statement	Defines numeric variables (and arrays) that hold values up to approximately sixteen significant digits.	Run or Immediate
BREAK	Statement	Exits from FOR-END, UNTIL-END, WHEN-END, and WHILE-END statements.	Run
CALL	Statement	Invokes an interface program.	Run or Immediate
CHAIN	Statement	Replaces the program currently in the work area or at the current DOLEVEL with another program and begins executing that program.	Run
CHR(*a*,...)	Function	Returns a text value consisting of the character(s) with ASCII code(s) "*a*" as specified.	Run or Immediate
CLEAR	Statement	Clears the Scroll Map Display, a fault trap, all program data, or data referred to by symbolic name.	Run or Immediate
COMMIT	Statement	Indicates the completion of a logical unit of work (LUW).	Run or Immediate
CONVERSE	Statement	Sends a formatted screen design to a terminal and returns the responses made by the operator.	Run or Immediate
COS(*a*)	Function	Returns the cosine of "*a*" where "*a*" is in radians.	Run or Immediate
CURSOR	Function	Determines the location of the cursor during the last terminal I/O.	Run or Immediate

Name	Types	Description	Mode
DATAFREE	Function	Returns the number of free bytes in the data work area.	Run or Immediate
DATE (function)	Function	Returns the text value of the current date.	Run or Immediate
DATE (statement)	Statement	Specifies the date mask to be used by the DATE function.	Run or Immediate
DECIMAL	Statement	Defines numeric variables (and arrays) that hold values up 31 significant digits.	Run or Immediate
DELETE	Statement	Deletes a record from a file or a row from a view.	Run or Immediate
DEQUEUE	Statement	Releases an enqueued resource.	Run or Immediate
DISPLAY	Statement	Displays the attributes (type, dimensions, value) or specified MANTIS variables, or displays the current MANTIS option values.	Run or Immediate
DO	Statement	Executes an internal or external subroutine.	Run or Immediate
DOLEVEL	Function	Returns your current level in an external subroutine.	Run or Immediate
E	Function	Returns the value of natural e.(2.71828183).	Run or Immediate
EDIT LIST	Statement	Invokes an external full-screen editor to edit a text variable or array.	Run or Immediate
EDITRCV LIST	Statement	Invokes an external full-screen editor to recover a previously aborted EDIT LIST session for a text variable or array.	Run or Immediate
ENQUEUE	Statement	Holds a resource.	Run or Immediate
ENTRY-EXIT	Statement	Defines the boundaries of a subroutine or program.	Run

Name	**Types**	**Description**	**Mode**
EXP(*a*)	Function	Returns the value of natural (e) to the power of "*a*."	Run or Immediate
FALSE	Function	Returns the value FALSE (0).	Run or Immediate
FILE	Statement	Identifies a MANTIS file that your program will access.	Run or Immediate
file-name	Function	Returns a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on a MANTIS file.	Run or Immediate
FOR-END	Statement	Repeats execution of a block of statements while a counter is incremented or decremented through a range of values.	Run
FORMAT(*a*,*m*)	Function	Returns a text-string conversion of a numeric expression according to a supplied edit mask.	Run or Immediate
FSI(*v*)	Function	Indicates the status of a GET, DELETE, INSERT, or UPDATE operation on an RDM user view, external file view, or MANTIS file.	Run or Immediate
GET	Statement	Reads a record from a file or a row from a view.	Run or Immediate
HEAD	Statement	Centers a heading on the top line of the Scroll Map Screen.	Run or Immediate
IF-ELSE-END	Statement	Executes a block of statements if a specified condition is true, another block if the condition is false.	Run
INSERT	Statement	Inserts a new record into a file or a row into a view.	Run or Immediate
INT(*a*)	Function	Returns the integer value of "*a*," truncating fractions toward zero.	Run or Immediate
INTEGER	Statement	Defines integral numeric variables (and arrays) that are signed 32-bit values.	Run or Immediate

Name	Types	Description	Mode
INTERFACE	Statement	Identifies an interface that your program will access.	Run or Immediate
interface-name	Function	Returns the text value stored in the status variable by the interface program.	Run or Immediate
KEY	Function	Returns a text value identifying your response to the most recent CONVERSE, OBTAIN, PROMPT, or WAIT statement.	Run or Immediate
LANGUAGE (statement)	Statement	Changes the current setting of your default language.	Run or Immediate
LANGUAGE (function)	Function	Returns the default language of the user who is currently signed-on.	Run or Immediate
LET	Statement	Assigns a value to a variable, array element, or text substring.	Run or Immediate
LOG(*a*)	Function	Returns the natural logarithm of "*a*."	Run or Immediate
$LOGICAL	Function	Allow access to logical names from a MANTIS program.	Run or Immediate
LOWERCASE(*t*)	Function	Returns the text value of "*t*" with any uppercase letters changed to lowercase.	Run or Immediate
MARK	Statement	Saves the current position of an RDM user view established by the most recent GET, UPDATE, INSERT, or DELETE statement.	Run or Immediate
MODIFIED(*s*,*f*)	Function	Tests whether a specific field or number of fields within a map definition changed during the last terminal I/O.	Run or Immediate
NEXT	Statement	Executes the next repeat in FOR-END, UNTIL-END, or WHILE-END statements or the next condition in a WHEN-END statement.	Run

Name	Types	Description	Mode
NOT(a)	Function	Returns TRUE(1) if "*a*" evaluates to FALSE(0);otherwise, returns FALSE(0).	Run or Immediate
NULL	Function	Returns a zero length text value.	Run or Immediate
NUMERIC	Function	Determines if a text expression contains a valid number.	Run or Immediate
OBTAIN	Statement	Obtains data from the Input Map and assigns input data to arithmetic and/or text variables.	Run or Immediate
$OPTION(*t*)	Function	Returns the value of a MANTIS option.	Run or Immediate
OUTPUT	Statement	Routes output from CONVERSE or SHOW statements to the screen, to the printer, or both.	Run or Immediate
PAD	Statement	Inserts padding characters into a text variable, array element, or substring.	Run or Immediate
PASSWORD	Function	Returns a text value containing the password entered during MANTIS sign-on.	Run or Immediate
PERFORM	Statement	Invokes an external command.	Run or Immediate
PI	Function	Returns the value of Pi (3.14159265358979).	Run or Immediate
POINT(*t*+*t*)	Function	Returns a number representing the position where a string addition or subtraction would occur if you executed it.	Run or Immediate
PRINTER (statement)	Statement	Assigns a new printer value.	Run or Immediate
PRINTER (function)	Function	Returns a text value containing the current printer assignment.	Run or Immediate
PROGFREE	Function	Returns the number of free bytes in the program work area.	Run or Immediate

Name	Types	Description	Mode
PROGRAM	Statement	Defines an external subroutine for your program to use.	Run or Immediate
PROMPT	Statement	Displays a prompter.	Run or Immediate
RELEASE	Statement	Releases RDM internal storage, closes an external file, or releases internal storage used by an external subroutine.	Run or Immediate
RESET	Statement	Backs out a logical unit of work (LUW).	Run or Immediate
RETURN	Statement	Returns control from a subroutine or stops execution of a program.	Run
RND(*a*)	Function	Returns a random real number in the range zero to “*a*,” but excluding zero and “*a*.”	Run or Immediate
SCREEN	Statement	Defines a screen design your program will use.	Run or Immediate
screen-name	Function	Returns a text value identifying your response to the most recent CONVERSE statement for the screen.	Run or Immediate
SCROLL	Statement	Specifies the amount by which a windowing terminal function will move the terminal window.	Run or Immediate
SEED	Statement	Seeds the random number generator to generate a new sequence of random numbers.	Run or Immediate
SET	Statement	Sets a fault trap, assigns a value to a symbol, or sets a logical name.	Run or Immediate
SGN(*a*)	Function	Returns -1 if *a*<0; 0 if *a*=0; +1 if *a*>0.	Run or Immediate
SHOW	Statement	Displays data or a fault trap in scroll mode.	Run or Immediate

Name	Types	Description	Mode
SIN(*a*)	Function	Returns the sine of “*a*” where “*a*” is in radians.	Run or Immediate
SIZE	Function	Returns the size and dimensions of an expression, variable, or array.	Run or Immediate
SLICE	Statement	Specifies the number of statements in a program slice.	Run or Immediate
SLOT	Statement	Specifies the number of program slices to be executed before a loop warning.	Run or Immediate
SMALL	Statement	Defines numeric variables (and arrays) that hold values up to approximately seven significant digits.	Run or Immediate
SQR(*a*)	Function	Returns the square root of “*a*.”	Run or Immediate
STATS	Statement	Allows your MANTIS program to extract environment process statistics.	Run or Immediate
STOP	Statement	Stops program execution.	Run or Immediate
$SYMBOL(*t*)	Function	Returns the text value assigned to a symbol.	Run or Immediate
TAN(a)	Function	Returns the tangent of “*a*” where “*a*” is in radians.	Run or Immediate
TERMINAL	Function	Returns a text character string of 1–8 characters containing the terminal ID.	Run or Immediate
TERMSIZE	Function	Returns a text value giving the terminal size in rows and columns.	Run or Immediate
TEXT	Statement	Defines text variables and arrays of text variables.	Run or Immediate
TIME (statement)	Statement	Specifies the time mask to be used by the TIME function.	Run or Immediate
TIME (function)	Function	Returns a text value expressing the current system time.	Run or Immediate

Name	Types	Description	Mode
TRAP	Statement	Intercepts certain error conditions during access to a MANTIS file, external file, ULTRA File, or and RDM user view.	Run or Immediate
TRUE	Function	Returns the value +1.	Run or Immediate
TXT(*a*)	Function	Returns the text value of “*a*,” in numeric display format.	Run or Immediate
ULTRA	Statement	Defines an ULTRA file view.	Run or Immediate
ultra-name	Function	Returns a text value indicating the status of the most recent GET, INSERT, UPDATE, or DELETE operation performed on an ULTRA file.	Run or Immediate
UNPAD	Statement	Removes padding characters from a text variable, array element, or substring.	Run or Immediate
UNTIL-END	Statement	Repeats execution of a block of statements until a specified condition becomes true.	Run
UPDATE	Statement	Replaces a record on a file or a row in a view with an updated record or row.	Run or Immediate
UPPERCASE(*t*)	Function	Returns the text value of “*t*” with any lowercase letters changed to uppercase.	Run or Immediate
USER	Function	Returns a 1–16 character text value containing the user name entered during MANTIS sign-on.	Run or Immediate
USERWORDS	Function	Returns the number of MANTIS symbolic names currently in use.	Run or Immediate
VALUE(*t*)	Function	Returns the numeric value of the text expression “*t*.”	Run or Immediate

Name	Types	Description	Mode
VIEW	Statement	Defines an RDM user view, or signs on to or off from RDM.	Run or Immediate
view-name	Function	Returns a text value indicating the status of the most recent GET, INSERT, UPDATE, or DELETE operation performed on an RDM user view.	Run or Immediate
VSI( )(*v*)	Function	Indicates the highest status of a field in an RDM user view.	Run or Immediate
WAIT	Statement	Temporarily suspends execution of your program until you press RETURN or a reply key sequence.	Run or Immediate
WHEN-END	Statement	Executes a block of statements only when a specified condition is true.	Run
WHILE-END	Statement	Repeats execution of a block of statements while a specified condition is true.	Run
ZERO	Function	Returns the value zero.	Run or Immediate

The rest of this chapter contains all of the MANTIS statements and functions in alphabetical order.

ABS

Use the ABS function to return the absolute value of an arithmetic expression.

ABS(*a*)

a

Description *Required*. Specifies the arithmetic expression whose absolute value you want returned.

Format Arithmetic expression

General consideration

- ♦ See also “SGN” on page 333.

Examples

```
ABS(0)        returns    0
ABS(-14E9)    returns    14E9
ABS(-.5)      returns    .5
```

ACCESS

Use the ACCESS statement to define the external file view that your program accesses. MANTIS retrieves the external file view from the user library and defines MANTIS variables in the work area for each field in the view.

ACCESS *access – name(libname,password*[*PREFIX*][*, levels*]

[*, NEW*]

[*, REPLACE*][*, FILE extname*

access-name

Description	*Required*. Specifies the symbolic name of the external file view for use in subsequent GET, UPDATE, INSERT, DELETE, RELEASE, and TRAP statements.
Format	A MANTIS symbolic name (See "MANTIS symbolic names" on page 65)
Consideration	No processing occurs if *access-name* is already defined.

libname

Description *Required*. Specifies the library name of the external file view as specified during External File View Design.

Format Text expression that evaluates to a library name in the format:

```
[user-name:] external-file-view-name
```

Consideration If the external file view is in your own library, you do not need to specify *user-name*.

password

Description *Required*. Specifies the password needed for the type of access (READ, UPDATE, INSERT/DELETE, SHARED UPDATE) your program requires.

Format Text expression that evaluates to the password specified during External File View Design.

Considerations

- ♦ MANTIS opens the file (or another record stream on the file) with an access-mode request based upon the level specified by the password. The file sharing access mode, which controls access to the file by other external processes, is also determined by the password as follows:
- ♦ READ: Read-only access-mode with shared-write (that is, you can only read the file, but other external processes may update it).
- ♦ UPDATE, INSERT/DELETE: Read/write access-mode with shared-read (that is, you may update the file, but other external processes can only read it).
- ♦ SHARE: Read/write access-mode with shared-write (that is, everybody may update the file).
- ♦ If there are multiple accesses in an application on one file, the access-mode level of subsequent accesses may not exceed that specified for the first access (that is, if the first ACCESS is specified as READ ONLY, the second ACCESS is restricted to READ ONLY, even if it specifies UPDATE). Make sure your application is set up so that the maximum level of access is specified in the first ACCESS.
- ♦ Alternatively, you can use the RELEASE statement to close the file, before executing more ACCESS statements that specify exclusive access-modes. The file is reopened with the requested access-mode, unless other users of the file already have access that locks you out. Note that the RELEASE statement can be used to temporarily close a file, which is reopened if necessary on the next GET, UPDATE, INSERT, or DELETE statement on that file.

PREFIX

Description *Optional*. Tells MANTIS to put the prefix “*access-name_*” on all variable names associated with this external file view. For example, if the external file view BOTTLES has a field named VOLUME, the following statement causes the corresponding variable BIN_VOLUME to be defined in the work area:

```
10 ACCESS BIN("BOTTLES","MAGIC",PREFIX)
```

Consideration MANTIS also prefixes the reference variable specified in the external file view for a sequential or relative file.

levels

Description *Optional*. Specifies the number of levels you want to use in GET, UPDATE , INSERT, or DELETE statements for the external file.

Format Arithmetic expression that evaluates to a value in the range 1–255

Considerations

- If you specify *levels*, you must also specify LEVEL=*level-number* (unless you want the default LEVEL=1) in all GET, UPDATE, INSERT, and DELETE statements for the external file.
- If you do not specify *levels*, MANTIS defines an INTEGER, SMALL, BIG, DECIMAL, or TEXT variable or array for each field in the external file profile according to the type and dimensions specified during External File View Design.
- If you specify *levels*, MANTIS adds an extra dimension to each field to allow a separate value to be stored at each level. MANTIS defines a one-dimensional array instead of a variable, a two-dimensional array instead of a one-dimensional array, and so on. The first dimension of each array is *levels*.
- For example, a field X defined as a four-byte float field with dimensions of 10 and 20 is equivalent to SMALL X(10,20). If the ACCESS statement specifies *levels* as 16, then it is equivalent to SMALL X(16,10,20).

NEW

Description	*Optional*. Creates a new version of the external file.
Consideration	The file description is obtained from the FDL file specified during External File View Design. MANTIS uses a default FDL file identified by the logical name MANTIS_CREATE if an FDL was not specified during External File View Design.

REPLACE

Description	*Optional*. Replaces the existing version of the external file.

FILE *extname*

Description *Optional*. Specifies an external file name that is used instead of the name provided in the external file view.

Format Text expression that evaluates to a valid file specification (or logical name that identifies an external file).

General considerations

- By default, unless you specify the NEW parameter, the external file must already exist. Otherwise MANTIS will fault because the file open will fail. This behavior is determined by ACCESS Options set in the External File View Design Facility, Library Functions. You may alter the Options so that MANTIS creates the external file if it doesn't exist. For the NEW parameter, you may alter the corresponding ACCESS Option so that MANTIS faults if the external file already exists.
- **OpenVMS** If the REPLACE parameter is specified, MANTIS will fail to replace the external file if it is currently open for write by another process.

Examples

```
20 .ACCESS RECORD("INDEX","SERENDIPITY",16)
30 .SCREEN MAP("INDEX")
40 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
50 ..CLEAR MAP: LEVEL_NUMBER=1
60 ..GET RECORD LEVEL=LEVEL_NUMBER
100 .ACCESS NEW_RECORD("NEW_INDEX","CONTENT",NEW)
```

access-name

Use the access-name function to return a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on an external file using the symbolic name “*access-name*.”

access-name

access-name

Description *Required*. Specify the symbolic name of the external file view.

Format Unique MANTIS symbolic name as defined in an ACCESS statement

Example

```
20 .ACCESS RECORD(INDEX,SERENDIPITY,16)
30 .SCREEN MAP("INDEX")
40 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
50 ..CLEAR MAP: LEVEL_NUMBER=1
60 ..GET RECORD LEVEL=LEVEL_NUMBER
SHOW RECORD
```

ASI (statement)

Use the ASI statement to specify a missing value for a field for RDM user views.

NOTE

This statement applies to SUPRA RDM users only.

ASI(*view-name*,*field-name*) = *text-expression*

view-name

Description	*Required*. Specifies the symbolic name by which you refer to the user view.
Format	MANTIS symbolic name as defined in a VIEW statement

field-name

Description	*Required*. Specifies the name of a particular field in the RDM user view.

text-expression

Description	*Required*. Specifies a text expression for the field.
Options	MISSING
	null (that is, an empty string)

Considerations

- ♦ If the text expression is equal to null or “MISSING,” the field is set to null on the next insert or update operation. All other values for text expressions are ignored.
- ♦ Note that any operation on an RDM user view (for example, INSERT, GET) results in the ASI for each field being modified. Consequently, if you want to perform a sequence of operations invoking null field values on an RDM user view, then the ASI for each null field must be set by the MANTIS program before each operation is performed.

- Note also that multilevel support is not available for null fields. If an RDM user view is declared with multiple levels (using the LEVEL specification in the VIEW statement), then the ASI status for each field must be inspected after each GET operation on the RDM user view. The ASI status cannot be maintained for each level declared in the VIEW statement.

- You must set up the field in SUPRA enabling null support before you use it in MANTIS; otherwise, MANTIS returns an error.

ASI (function)

Use the ASI function to indicate the status of each field in an RDM user view.

This statement applies to SUPRA RDM users only.

ASI(*view-name*,*field-name*)

view-name

Description *Required*. Specifies the symbolic name by which you refer to the user view.

Format MANTIS symbolic name defined in a previously executed VIEW statement.

field-name

Description *Required*. Specifies the name of a particular field in an RDM user view.

General consideration

- Attribute Status Indicators (ASIs) reflect the status of each field defined in your RDM user view. See the “DBCS support feature” on page 455 for more details on ASIs and user views. See also “FSI” on page 220 and “VSI( )” on page 396.

Example

```
20 VIEW PARTS("PARTS_ON_ORDER")
.
.
.
100 GET PARTS
110 IF ASI(PARTS,PART_NAME)="MISSING"
```

ATN

The ATN (arctangent) function returns the angle in radians whose tangent is an arithmetic expression (a).

ATN(*a*)

a

Description *Required*. Specifies the arithmetic expression whose arctangent angle you want returned.

Format Arithmetic expression

Considerations

- The expression result is coerced to BIG. Decimal expression results may therefore lose some precision or may generate an arithmetic fault (310) if the magnitude is too large for a BIG data type.
- The value returned will be between π/2 and -π/2.
- See also "TAN" on page 352, "COS" on page 167, "SIN" on page 338, and "PI" on page 303.

Examples

```
ATN(10)   returns    1.4711276743
ATN(100)  returns    1.5607966601
```

ATTRIBUTE (statement)

ATTRIBUTE is both a statement and a function. Use the ATTRIBUTE statement in your programs to change the attributes of a screen, a field on a screen, a terminal, or a printer. The changed attributes remain in effect until you change them again or use the RESET attribute to revert to the original specification. Use the ATTRIBUTE function at the command line to check the current status of field-level, map-level, terminal, and printer attributes. Both the ATTRIBUTE statement and function are described in the following pages.

$$\textbf{ATTRIBUTE}\left\{\begin{array}{l} \left(screen-name\begin{bmatrix},field-name\\ ,(lrow,lcol)\end{bmatrix}\right)=attributes,\ldots \\ \textbf{(TERMINAL)}=\textbf{[}page\text{-}size,\textbf{]}\,attributes,\ldots \\ \textbf{(TERMINAL,CURSOR)}=position \\ \textbf{(PRINTER)}=attributes,\ldots \end{array}\right\}$$

$$\textbf{ATTRIBUTE}\left(screen-name\begin{bmatrix},field-name\\ ,(lrow,lcol)\end{bmatrix}\right)=attributes,\ldots$$

screen-name

Description *Required*. Specifies the name (as defined in a previously executed SCREEN statement) of a screen design

Format MANTIS symbolic name defined in a previously executed SCREEN statement

field-name

Description *Optional*. Specifies the field whose attributes you want to change.

Format Symbolic name of a field in the screen as specified during Screen Design

(lrow,lcol)

Description *Optional*. Specifies the coordinates of the field whose attributes you want to change. These are the row and column coordinates of the field within the logical display. The selected field is the one in the active map that contains these coordinates.

Format *lrow* and *lcol* must each be a numeric expression that evaluates to a value from 1 to 32767

attributes

Description *Required*. Specifies which attributes you want to apply.

Format Text expression that evaluates to one or more attributes, with multiple attributes separated by commas, (for example, “CUR,PRO”)

Considerations

- ♦ If you do not specify *field-name*, field-level attributes (except CURSOR and MESSAGE) apply to all data fields in the specified screen. You can also specify attributes that apply to the screen itself. These map-level attributes are listed below, paired by opposites. For more information, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.

ALARM	NO ALARM
PROTECTED BOTTOM LINE	BOTTOM LINE UNPROTECTED
FULL DISPLAY	NO FULL DISPLAY

- ♦ Two other map-level attributes are available. They are KEEP MAP MODIFIED (KMM) and RESET MAP MODIFIED (RMM). KMM prevents MANTIS from clearing the modified data tags of the specified screen. MANTIS ordinarily clears the modified data tags of all maps in the mapset for a CONVERSE UPDATE or for the active map in the case of a CONVERSE SET. If the modified data tags were cleared, a previously modified map returns FALSE for the MODIFIED function. If a map has the attribute KMM, then once modified, the MODIFIED function returns TRUE. RMM will reset the modified data tag usage to the default.

See the discussion of the MANTIS Logical Terminal Interface in "Programming techniques" on page 415 for detailed examples of CONVERSE.

ATTRIBUTE (TERMINAL), $\left[\begin{matrix}\textit{page}\\ \textit{size}\end{matrix}\right]$ *,attribute,…*

TERMINAL

Description	*Required*. Specifies the MANTIS output device. This is the terminal screen when MANTIS is running interactively.

page-size

Description	*Optional*. Specifies the page size of the output device. This is the number of rows and columns that MANTIS outputs to the TERMINAL as one full screen.
Format	Text expression evaluating to a comma-separated row and column specification in the format: (*prow*,*pcol*)
Consideration	Values used must be less than or equal to the physical terminal screen size, to avoid unpredictable results.

attributes

Description *Required*. Specifies which attribute(s) you want to apply to the terminal.

Format A text expression that evaluates to one or more attributes, with multiple attributes separated by commas, (for example, “NOU,NOC”)

Options

BRIGHT	NORMAL
REVERSE VIDEO	VIDEO
REVERSE FULL FIELD	
BLINK	NO BLINK
UNDERLINE	NO UNDERLINE
UNDERLINE FULL FIELD	
CLASS	
DOUBLE HEIGHT	SINGLE HEIGHT
DOUBLE WIDTH	SINGLE WIDTH
COLOR	NOCOLOR

Considerations

- ♦ You are not normally required to set the TERMINAL attributes because MANTIS determines the type of the output device and attempts to find all the attributes of the device. However, you may wish to suppress some to these attributes, or override them.
- ♦ **OpenVMS** If there is a Terminal Capabilities entry for the output device in the OpenVMS TERMTABLE used by MANTIS, then all the attributes of the device are known. When searching for a Terminal Capabilities file entry, OpenVMS will first search TERM$TABLOC:TERMTABLE.EXE and, failing that, SYS$SYSTEM:TERMTABLE.EXE.
- ♦ **OpenVMS** The OpenVMS runtime library routines for screen management, SMG, are used by MANTIS to allow screen refresh optimization. Under certain situations it may be necessary to not use SMG, and thus not have the screen refresh optimization.

- **OpenVMS** Turn OFF use of SMG screen refresh optimization using ATTRIBUTE(TERMINAL)="CLASS(TERMINAL-TYPE)," where TERMINAL-TYPE is the name of the termtable entry. Generally this will be VT100, VT200_SERIES, VT300_SERIES, and so on. Turn use of SMG screen refresh optimization back ON using ATTRIBUTE(TERMINAL)="CLASS()."
- **OpenVMS** The MANTIS output device is identified by the logical name MANTIS_OUTPUT.
- **UNIX** The MANTIS output device is standard output, which is normally the terminal, but may be a file if output redirection has been used when MANTIS is invoked.

ATTRIBUTE (TERMINAL,CURSOR)=*position*

TERMINAL

Description *Required*. Specifies the MANTIS output device. This is usually the screen on which MANTIS is running.

CURSOR

Description *Required*. Specifies the position of the cursor in the physical screen display. The assigned position gives the row and column position where the cursor is to appear when the next terminal input operation begins, that is, upon executing the next OBTAIN or CONVERSE statement or command.

Consideration See also the "CURSOR" on page 168.

position

Description *Required*. Specifies the physical screen position in row/column coordinate form.

Format Text expression evaluating to a comma-separated row and column position in the format: (*prow*,*pcol*)

Consideration Values used must be less than or equal to the physical terminal screen size, to avoid unpredictable results.

ATTRIBUTE (PRINTER)=*attributes*

PRINTER

Description *Required*. Indicates that you want to change the attributes of the printer.

attributes

Description *Required*. Specifies the attributes you want to apply to the printer.

Format A text expression that evaluates to one or more attributes, with multiple attributes separated by commas

Considerations

- If you specify PRINTER, you must also specify attributes that apply to the printer device itself. These device-level attributes are listed below, paired by opposites.

CLASS	
SPOOL ON CLOSE	NO SPOOL ON CLOSE

- The CLASS attribute applies only to the OpenVMS environment. The attribute requires a terminal name in parentheses, for example, “CLASS(ANADEX).” The terminal name is the name of a type of terminal as defined in a TERMTABLE entry in a OpenVMS Terminal Capabilities File. MANTIS processes a CLASS attribute by obtaining device attributes from the specified TERMTABLE entry. TERMTABLE entries may have been set up for nonstandard printers at your installation. Your Master User should know whether you need to use this attribute.

- You can also specify the following attributes to indicate whether you want MANTIS to simulate them on a device that does not support them directly:

BRIGHT	NORMAL
UNDERLINE	NO UNDERLINE

- By default, MANTIS simulates BRIGHT and UNDERLINE if your printer does not support them. Specify NORMAL or NO UNDERLINE if you want to stop MANTIS from simulating these attributes.

- See also “OUTPUT PRINTER” on page 293.

- You can specify the size of a printed page in rows and columns as in the following example:

```
ATTRIBUTE(PRINTER)="(60,132)"
```

General consideration

- The following table alphabetically lists all the attributes, paired by opposites, that are recognized by the ATTRIBUTE statement (see the following section on the ATTRIBUTE function for the list of attributes that are returned by that function). For each attribute, the table indicates the type of entity it applies to (TERMINAL, PRINTER, screen or field) and whether the attribute may be set in Screen Design. Some attributes may be applied to more than one type of entity. (The screen design attributes are discussed further in the section on the ATTRIBUTE function, and are specified in the Update Field Specifications option of Screen Design as documented in *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300*.)*

Attribute Table

The following table displays the attributes and where they may be used, i.e. the Autoskip/Noautoskip may be used on a screen, in a field or during screen design.

Attribute	Terminal	Printer	Screen	Field	Screen design
AUTOSKIP / NO AUTOSKIP			X	X	X
BLINK / NO BLINK	X		X	X	X
BOXED / UNBOXED			X	X	X
BRIGHT / NORMAL / HIDDEN	X	X	X	X	X
CLASS	X	X			
COLOR/NO COLOR	X	X	X	X	
CURSOR / NO CURSOR			X	X	X
DETECTABLE / NON DETECTABLE	X		X	X	X
DOUBLE HEIGHT / SINGLE HEIGHT	X	X		X	X
DOUBLE WIDTH / SINGLE WIDTH	X	X		X	X
FULL DISPLAY / NO FULL DISPLAY			X		X
HIGHLIGHT / NO HIGHLIGHT	X	X	X	X	X
KEEP MAP MODIFIED / RESET MAP MODIFIED			X		
LEFT BAR / NO LEFT BAR			X	X	X
MESSAGE / NO MESSAGE			X	X	X
MODIFIED / UNMODIFIED			X	X	X
NO COLOR / BLUE / GREEN / NEUTRAL / PINK / RED / TURQUOISE / YELLOW / C*nn*			X	X	X

Attribute	Terminal	Printer	Screen	Field	Screen design
OVERLINE / NO OVERLINE			X	X	X
PROTECTED BOTTOM LINE/ BOTTOM LINE UNPROTECTED			X		X
PROTECTED / UNPROTECTED			X	X	X
PROTECTED INPUT ONLY/			X	X	X
RESET			X	X	
REVERSE VIDEO / VIDEO	X	X	X	X	X
REVERSE FULL FIELD / VIDEO	X	X	X	X	X
RIGHT BAR / NO RIGHT BAR			X	X	X
(ROW,COL)	X	X			
SOUND ALARM / NO ALARM			X		X
SPOOL ON CLOSE / NO SPOOL ON CLOSE		X			
UNDERLINE / NO UNDERLINE	X	X	X	X	X
UNDERLINE FULL FIELD / NO UNDERLINE	X	X	X	X	X
UPPERCASE / LOWERCASE			X	X	X

- The BOXED, UNBOXED, DETECTABLE, NON DETECTABLE, OVERLINE, NO OVERLINE, LEFT BAR, NO LEFT BAR, RIGHT BAR, MODIFIED, and UNMODIFIED attributes have no effect. They are included in MANTIS for compatibility with MANTIS for the IBM mainframe.

- Specify an attribute using the underlined three-character abbreviation indicated in the preceding table. MANTIS ignores any additional characters. For example, you may specify BRIGHT as “BRI,” “BRIGHT” or “BRILLIG.”

- Use commas to separate multiple attributes in a single text expression, for example, “BRI,UNP, and AUT.”

♦ Each attribute you specify resets any opposite attribute that is in effect. In addition:

  - BRIGHT sets DETECTABLE and resets HIDDEN.
  - NORMAL resets BRIGHT and HIDDEN.
  - HIDDEN resets BRIGHT and DETECTABLE.
  - DETECTABLE resets HIDDEN.
  - UNDERLINE resets UNDERLINE FULL FIELD and vice versa.
  - REVERSE resets REVERSE FULL FIELD and vice versa.
  - CURSOR resets CURSOR on all other fields in the screen.
  - MESSAGE sets PROTECT. MESSAGE also resets MESSAGE and
  - PROTECT on all other fields in the screen that have the MESSAGE attribute.
  - NO MESSAGE resets PROTECT on a field with the MESSAGE attribute.

♦ If you specify two or more conflicting attributes, MANTIS uses the last one specified.

♦ You can also set many attributes in Screen Design. (Refer to the *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more details.)

♦ The RESET attribute resets all attributes of a screen field to the original settings specified in Screen Design. RESET has no effect on device or map-level attributes.

♦ Note that a list of the attributes returned by the ATTRIBUTE function appears in the next section.

♦ You can only change the attributes of screen heading fields by using the (*lrow*,*lcol*) coordinates of the heading within the logical display.

♦ MANTIS allows you to specify field color using IBM conventions (BLUE, GREEN, and so on) or by number in the color palette (C*nn*). Specifying colors and mapping IBM colors are both described in Chapter 2 of *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300. Color extensions appear in Appendix D of that manual.

- You cannot change the height and/or width of a field dynamically.
- The MESSAGE attribute can only be applied to a TEXT field.
- See also: “CURSOR” on page 168 and “MODIFIED” on page 283.
- The following list provides details about the attributes that can be set by the ATTRIBUTE statement. As in the preceding table, the attributes are listed by opposite pairs (if an opposite attribute exists) in alphabetical order. Each pair is followed by a description and the format. The underlined letters indicate the allowable abbreviation for each attribute. Any special considerations for using the attribute are described after the format.

AUTOSKIP

NO AUTOSKIP

Description Specifies if the cursor will skip automatically to the next unprotected data field when data is entered in the far right position of the field.

Format AUTOSKIP or NO AUTOSKIP

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.
- When the NO AUTOSKIP attribute is specified, the user must use the TAB key to advance to the next field.
- Autoskip is operative from a batch input stream only if the BATAUTOSKIP MANTIS Option is TRUE.

BLINK

NO BLINK

Description Specifies if you want the field to blink when it is displayed, or specifies that the TERMINAL can logically support this attribute.

Format BLINK or NO BLINK

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.
- The three attributes, UNDERLINE, BLINK, and REVERSE VIDEO as a group are considered to be HIGHLIGHT. If you are specifying these attributes, you only get a choice of one. Depending on how you have set ATTRIBUTE(TERMINAL), and what your hardware allows, only one of the attributes works. The field can have only one highlighting property specified by the extended field attribute (such as blink or reverse, but not both). MANTIS takes the precedence of the attributes in the order listed above. See also the HIGHLIGHT attribute.

BOXED

UNBOXED

Description This attribute has no effect. It is included in MANTIS for compatibility with MANTIS for the IBM mainframe. In that environment, it specifies if a box should be created around a field, or specifies that the TERMINAL can support this attribute.

Format BOXED or UNBOXED

Considerations

- ♦ This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.
- ♦ The BOXED attribute represents a combination of the attributes LEFT BAR, RIGHT BAR, OVERLINE and UNDERLINE.

BRIGHT

NORMAL

HIDDEN

Description Specifies the intensity of a field, or specifies that the TERMINAL can logically support this attribute.

Format BRIGHT, NORMAL, or HIDDEN

Consideration This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.

- ♦ See also the HIGHLIGHT attribute

CLASS

Description Specifies a terminal/device entry in a OpenVMS Terminal Capabilities File, that is, an entry in either the system TERMTABLE.EXE file and/or in a TERMTABLE.EXE file in the directory identified by the logical name TERM$TABLOC.

Format CLASS(*xxxx*), where *xxxx* must match a name specified in one of the 'NAME=' statements in a TERMTABLE.

Considerations

- ♦ This attribute is only applicable to the OpenVMS environment.
- ♦ By default, MANTIS obtains the TERMTABLE entry applicable to a given device (TERMINAL or PRINTER) according to the type of device returned by the $GETDVI system service (DVI$_DEVTYPE). Since the usual PRINTER destination is a file and not a device, you may want to use the CLASS attribute to obtain a file more suitable for printing on a given printer device.

COLOR

NO COLOR

Description Specifies whether the terminal can logically support color output, or turns off any color attribute currently set on a field.

Format COLOR or NO COLOR

Considerations

- ♦ For more information about color output, refer to Chapter 2 and Appendix C of *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.
- ♦ For more information about the color attributes that can be specified in Screen Design see the "Attribute Table" on page 118. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.

CURSOR

NO CURSOR

Description Indicates whether the cursor will be automatically positioned in the field when you converse the screen.

Format CURSOR or NO CURSOR

Considerations

- When more than one field on a screen has the CURSOR attribute set by Screen Design, the first one (in a left to right top to bottom order) will have the cursor positioned in it when the screen is conversed. If that first field is given the NO CURSOR (NCU) attribute, the next such field will contain the cursor. If there are no more such fields, MANTIS positions the cursor in the first unprotected data field.

- Using the statement ATTRIBUTE(*map*,*field*)="CUR" turns off the cursor for all fields in the map except the nominated field. Therefore, when the cursor has been set to multiple fields, the most recent one set will contain the cursor. Any other maps in the map set are not affected by this statement and any cursor settings for other maps will remain turned on.

- Using the statement ATTRIBUTE(*map*,*field*)="RES" or ATTRIBUTE(map)="RES" restores the original Screen Design values for cursor setting. (See the RESET attribute for more information).

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.

- If you specify more than one field containing the cursor, the last specification is used. ATTRIBUTE(TERMINAL,CURSOR) = "(*row*,*col*)" takes precedence over all field cursor specifications.

DETECTABLE

NON DETECTABLE

Description This attribute has no effect. It is included in MANTIS for compatibility with MANTIS for the IBM mainframe. In that environment, it specifies if a field on a screen will be pen detectable.

Format DETECTABLE or NON DETECTABLE

Consideration This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.

DOUBLE HEIGHT

SINGLE HEIGHT

Description Indicates whether a field should double its vertical size, or specifies whether the terminal or printer can logically support display of double-height characters.

Format DOUBLE HEIGHT or SINGLE HEIGHT

Considerations

♦ If you converse a screen that contains double height/width fields on a device that does not support this feature (or if the feature is disabled), MANTIS displays all fields with single height, and width format. No warnings are issued. This also occurs in IBM compatibility mode.

♦ If you disable double height for a device, double height fields are displayed as single height, double width. If you disable double width for a device, double height is implicitly disabled as well. As a result, fields with double height or double width attributes are displayed with single height, single width.

♦ This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.

DOUBLE WIDTH

SINGLE WIDTH

Description Indicates whether a field should double its horizontal size, or specifies whether the terminal or printer can logically support display of double-width characters.

Format DOUBLE WIDTH or SINGLE WIDTH

Considerations

- If you converse a screen that contains double height/width fields on a device that does not support this feature (or if the feature is disabled), MANTIS displays all fields with single height, and width format. No warnings are issued. This also occurs in IBM compatibility mode.

- If you disable double height for a device, double height fields are displayed as single height, double width. If you disable double width for a device, double height is implicitly disabled as well. As a result, fields with double height or double width attributes are displayed with single height, single width..

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.

FULL DISPLAY

NO FULL DISPLAY

Description Indicates whether MANTIS will expand the screen size to the dimensions of the current terminal, including the bottom two lines of the screen, which are normally reserved for the input map.

Format FULL DISPLAY or NO FULL DISPLAY

Considerations

- This attribute can also be specified in the Library Functions of Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.

- If you specify this attribute, the input map is not displayed (thus, for example, you cannot enter data into the unsolicited input and reply fields, or see error messages displayed in the input map’s message field). If you need to make use of any of the input map fields, you can use the INPUTMAP (GOLD/I) terminal function during the CONVERSE to toggle the display of the input map over the bottom two lines of the terminal screen.

HIGHLIGHT

NO HIGHLIGHT

Description Indicates that a field will be highlighted with any highlighting attribute supported by the terminal when it is displayed.

Format HIGHLIGHT or NO HIGHLIGHT

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.

- The HIGHLIGHT attribute should be used with one of the following attributes: BRIGHT, REVERSE VIDEO, BLINKING, or UNDERLINE. On a terminal without advanced video options, MANTIS will display the field using either UNDERLINE or REVERSE VIDEO instead of the nominated attribute, depending on the physical characteristics of the terminal.

KEEP MAP MODIFIED

RESET MAP MODIFIED

Description Prevents MANTIS from clearing the “modified data tags” of a specified screen.

Format KEEP MAP MODIFIED or RESET MAP MODIFIED

Considerations

- Specify KEEP MAP MODIFIED (KMM) to prevent MANTIS from clearing the “modified data tags” (or MDTs) of the specified screen. Ordinarily, MANTIS clears MDTs of all maps in the mapset for a CONVERSE UPDATE, or for the “active map” in the case of a CONVERSE SET. If the MDTs are cleared, a previously modified map returns FALSE for the MODIFIED function. If a map has the KMM attribute, then once modified, the MODIFIED function always returns TRUE. For an explanation of maps and map sets, see the “CONVERSE” statement on page 159.
- Specify RESET MAP MODIFIED (RMM) to turn off KMM and restore ordinary functionality.
- Use KMM if you will converse additional screens in the mapset (for example, pop-up screens) and want to retain the MODIFIED setting for further validation after the CONVERSE of the additional screens.

LEFT BAR

NO LEFT BAR

Description This attribute has no effect. It is included in MANTIS for compatibility with MANTIS for the IBM mainframe. In that environment, it specifies that a left bar appears on a field.

Format LEFT BAR or NO LEFT BAR

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.
- See also the BOXED attribute.

MESSAGE

NO MESSAGE

Description Indicate whether you want MANTIS to use this field for display of system messages (for example, field validation error messages). By default, MANTIS displays system messages in the Message Field of the converse input map.

Format MESSAGE or NO MESSAGE

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.
- If more than one field in a screen has the MESSAGE attribute, the field with the lowest row and column coordinates is considered the MESSAGE field for that screen when the screen is conversed.
- If the ATTRIBUTE(. . .)="MSG" statement is executed, the MESSAGE attribute will be canceled in all other fields in the screen. If more than one field is given the MESSAGE attribute by the ATTRIBUTE statement, only the last ATTRIBUTE statement is effective.
- When a system message is to be displayed during a CONVERSE operation, the display field is determined as follows:
    - The map set is searched for a field with the MESSAGE attribute that is visible on the terminal display. The Active Map (that is, the map most recently added to the map set) is searched first, and then the map(s) below it until a visible MESSAGE field is found.
    - If no visible MESSAGE field is found, the Message Field of the converse input map is used.

MODIFIED

UNMODIFIED

Description This attribute is included in MANTIS for compatibility with MANTIS for the IBM mainframe. In that environment it indicates that the field contents are sent to the Data Work Area when you press ENTER or a PF key.

Format MODIFIED or UNMODIFIED

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300 for more information.
- Specify the MODIFIED attribute if you will be using the screen in the MANTIS for the IBM mainframe environment and you want the field marked as MODIFIED each time you converse the screen.

NO COLOR

NEUTRAL/BLUE/ PINK/GREEN/TURQUOISE/RED/YELLOW/Cnn

Description Indicates the color of a field, or turns off any color attribute currently set on a field.

Format NO COLOR or NEUTRAL/BLUE/ PINK/GREEN/TURQUOISE/RED/YELLOW, C*nn*

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.
- For more information about color output, refer to Chapter 2 and Appendix C of *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.
- See also the COLOR attribute.

OVERLINE

NO OVERLINE

Description This attribute has no effect. It is included in MANTIS for compatibility with MANTIS for the IBM mainframe. In that environment, it specifies if a line appears over a field.

Format OVERLINE or NO OVERLINE

Considerations

- ♦ This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.
- ♦ See also the BOXED attribute.

PROTECTED BOTTOM LINE

BOTTOM LINE UNPROTECTED

Description Indicates whether you want MANTIS to protect the bottom line of a screen (Unsolicited Input and Reply Field) from input.

Format PROTECTED BOTTOM LINE or BOTTOM LINE UNPROTECTED

Considerations

- ♦ This attribute can also be specified in the Library Functions of Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.
- ♦ See also the FULL DISPLAY attribute.

PROTECTED

UNPROTECTED

Description Indicates whether you want a field protected from both data entry and tabbing into the field.

Format PROTECTED or UNPROTECTED

Consideration This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.

PROTECTED INPUT ONLY

UNPROTECTED

Description Indicates whether you want a field protected against input only (data cannot be entered in the field but the field may be tabbed into)

Format PROTECTED INPUT ONLY or UNPROTECTED

Consideration This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.

RESET

Description Resets all attributes of a screen field to the original settings specified in Screen Design. RESET has no effect on map-level attributes.

Format RESET

Consideration Using the statement ATTRIBUTE(*map*,*field*)=“RES” or ATTRIBUTE(*map*)=“RES” restores the original Screen Design values for cursor setting. (See the CURSOR/NO CURSOR attribute for more information.)

REVERSE VIDEO

VIDEO

Description Indicates whether the data in a field displays in reverse video, or specifies whether the TERMINAL or PRINTER logically supports this attribute.

Format REVERSE VIDEO or VIDEO

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.
- See also the REVERSE FULL FIELD and HIGHLIGHT attributes.

REVERSE FULL FIELD

VIDEO

Description Indicates whether the entire field (not just the data in the field) displays in reverse video, or specifies whether the TERMINAL or PRINTER logically supports this attribute.

Format REVERSE FULL FIELD or VIDEO

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.
- See also the REVERSE VIDEO and HIGHLIGHT attributes.

RIGHT BAR

NO RIGHT BAR

Description This attribute has no effect. It is included in MANTIS for compatibility with MANTIS for the IBM mainframe. In that environment, it specifies that a right bar appears on a field.

Format RIGHT BAR or NO RIGHT BAR

Consideration This attribute can also be specified using the BOX attribute in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.

SOUND ALARM

NO ALARM

Description Specifies whether you want the terminal to beep each time the screen is CONVERSEd.

Format SOUND ALARM or NO ALARM

Consideration This attribute can also be specified in the Library Functions of Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.

SPOOL ON CLOSE

NO SPOOL ON CLOSE

Description Indicates whether your PRINTER file is automatically spooled when it is closed by MANTIS.

Format SPOOL ON CLOSE or NO SPOOL ON CLOSE

Considerations

- ♦ By default, your PRINTER is set up to be a file which is automatically spooled when the file is closed.
- ♦ Your PRINTER setting is originally taken from your user profile during MANTIS sign-on, but can be superseded at any time by the PRINTER statement. Refer to the PRINTER statement for more information.
- ♦ **OpenVMS** Spooling is either to SYS$PRINT or to a specified printer queue if '/QUEUE=' is specified in your PRINTER assignment.
- ♦ **OpenVMS** MANTIS has to use a OpenVMS subprocess to print the file if a printer QUEUE is specified in the PRINTER assignment. In this case the value of PRINTER is appended to the value of the PRINTCMD MANTIS Option and the resulting command is executed in a subprocess.
- ♦ **UNIX** The PRINTCMD MANTIS Option specifies the spool command to use to print the file.
- ♦ **UNIX** The filename specified in PRINTER is appended to the value of the PRINTCMD MANTIS Option and MANTIS forks a child process to execute the resulting command.

UNDERLINE

NO UNDERLINE

Description Indicates whether the data in a field is underlined when it displays, or specifies whether the TERMINAL or PRINTER logically supports this attribute.

Format UNDERLINE or NO UNDERLINE

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.
- See also the UNDERLINE FULL FIELD and HIGHLIGHT attributes.

UNDERLINE FULL FIELD

NO UNDERLINE

Description Indicates whether the entire field (not just the data in the field) is underlined when it displays, or specifies whether the TERMINAL or PRINTER logically supports this attribute.

Format UNDERLINE FULL FIELD or NO UNDERLINE

Considerations

- This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.
- See also the UNDERLINE and HIGHLIGHT attributes.

UPPERCASE

LOWERCASE

Description Indicates whether MANTIS will convert input to uppercase or not (left as entered)

Format <u>UPP</u>ERCASE or <u>LOW</u>ERCASE

Consideration This attribute can also be specified in Screen Design. Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300, for more information.

Examples The following example sets the ACCT_NUM field associated with the screen INVOICE to BRIGHT and PROTECTED:

```
10 SCREEN INVOICE ("INVOICE")
20 TEXT STANDARD,TEMP
30 STANDARD="BRIGHT,PROTECTED"
40 ATTRIBUTE(INVOICE,ACCT_NUM)=STANDARD
```

The following examples produce the same result as the previous example:

```
30 STANDARD="BRIGHT"
40 ATTRIBUTE(INVOICE,ACCT_NUM)=STANDARD+
50 '",PROTECTED"
30 STANDARD="BRI"
40 ATTRIBUTE(INVOICE,ACCT_NUM)=STANDARD,"PRO"
```

The following example sets the ACCT_NUM field associated with the screen INVOICE to BRIGHT, PROTECTED, and CURSOR:

```
 30 STANDARD="PROTECTED"
 40 TEMP="CURSOR"
 50 ATTRIBUTE(INVOICE,ACCT_NUM)="BRIGHT"
 .
 .
 .
160 ATTRIBUTE(INVOICE,ACCT_NUM)=STANDARD
 .
 .
 .
240 ATTRIBUTE(INVOICE,ACCT_NUM)=TEMP
```

The following example produces the same result as the previous example:

```
240 ATTRIBUTE(INVOICE,ACCT_NUM)="PRO,BRI,CUR"
```

The following example resets all attributes to their original specifications:

```
240 ATTRIBUTE(INVOICE)="RESET"
```

This example shows how to use the ATTRIBUTE(TERMINAL) and ATTRIBUTE(PRINTER) statements to set values.

```
 10 ATTRIBUTE(TERMINAL)="COLOR"
 20 ATTRIBUTE(PRINTER)="CLASS(LN03),(66,132)"
```

The following example places the cursor at location (15,15) on the physical terminal.

```
 10 ATTRIBUTE(TERMINAL,CURSOR)="(15,15)"
```

ATTRIBUTE (function)

The ATTRIBUTE function returns attributes of a field or device or the physical coordinates of the cursor on the screen.

$$\textbf{ATTRIBUTE}\left\{\begin{array}{l} (\textit{screen - name} \left[\begin{array}{l}\textbf{,}\,\textit{field - name}\\ \textbf{, (}\textit{lrow, lcol}\textbf{)}\end{array}\right]) \\ (\textbf{TERMINAL}) \\ (\textbf{TERMINAL,CURSOR}) \\ (\textbf{PRINTER})\end{array}\right\}$$

screen-name

Description *Required*. Specify the name of the screen design as defined in the SCREEN statement.

Format A MANTIS symbolic name defined in a previously executed SCREEN statement.

field-name

Description *Optional*. Specify the name of the field whose attributes you want returned.

Format Valid symbolic name of a field in the screen as specified during Screen Design.

(lrow,lcol)

Description *Optional*. Specify the coordinates of the field in the logical display whose attributes you want returned.

Considerations

- ♦ The row and column positions must fall within a field in the specified screen, or MANTIS issues an error message.
- ♦ For double-height fields, you must specify the logical display coordinates of the lower half of the field.

General consideration

- ♦ For a field, attributes are returned in the following format:

```
(row,col),length,attributes
```

where (*row*,*col*) specifies the coordinates of the field in the screen design, *length* specifies the length of the field, and *attributes* specifies the current attribute settings for the field, for example: BRI,PRO.

ATTRIBUTE (TERMINAL)

TERMINAL

Description *Required*. Indicates that you want MANTIS to return the attributes of your terminal.

General consideration

- ♦ Attributes are returned as text in the following form:

```
(prow,pcol),attributes
```

where *prow* and *pcol* are the row and column positions of the cursor in the physical screen display.

ATTRIBUTE(TERMINAL,CURSOR)

TERMINAL,CURSOR

Description *Required*. Indicates that you want MANTIS to return the position of the cursor in the physical screen display.

Consideration Attributes are returned as text in the following form:

```
(prow,pcol)
```

where *prow* and *pcol* are the row and column positions of the cursor in the physical screen display.

ATTRIBUTE(PRINTER)

PRINTER

Description *Required*. Indicate that you want MANTIS to return the attributes of your printer.

Consideration Attributes are returned as text in the following form:

```
(prow,pcol),attributes
```

where *prow* and *pcol* are the number of rows and columns which MANTIS will output to each page directed to the PRINTER.

General consideration

- The ATTRIBUTE function returns the attributes of the field, screen, terminal, or printer you are testing. The following lists the attributes that can be returned, in alphabetical order. The underlining indicates the abbreviation that MANTIS will use to return the attribute, for example, “BLI” for a field that is set to “BLINK.” For more information on the attributes, see the “Attribute Table” on page 118, and refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.

AUTO SKIP	FIELD SENSITIVE VALIDATION	NO HORIZONTAL WINDOWING	REVERSE VIDEO
BLINK	FILL	NO SPOOL ON CLOSE	RIGHT BAR
BOTTOM LINE UNPROTECTED	FORCED FIELD SENSITIVE VALIDATION	NO VERTICAL WINDOWING	SHOW DEFAULT
BOXED	FULL DISPLAY	NUMERIC	SOUND ALARM
BRIGHT	HEADING	NUMERIC MASK	SPOOL ON CLOSE
COLOR C00 ...C15 BLUE RED PINK GREEN TURQUOISE YELLOW NEUTRAL	HIDDEN	OPAQUE	UNDERLINE
CURSOR	HIGHLIGHT	OVERLINE	UNDERLINE FULL FIELD
DEFAULT VALUE	KANJI	PROTECTED	UPPERCASE
DETECTABLE	KEEP MAP MODIFIED	PROTECTED BOTTOM LINE	VALID NAME
DEVICE CLASS	LEFT BAR	PROTECTED INPUT ONLY	VALID VARIABLE
DOUBLE HEIGHT	MESSAGE	RANGE CHECK	VALIDATION LIST
DOUBLE WIDTH	MODIFIED	REQUIRED	WINDOWING
FIELD ENTRY ROUTINE	NO ALARM	RESET MAP MODIFIED	
FIELD EXIT ROUTINE	NO FULL DISPLAY	REVERSE FULL FIELD	

Examples

Function specification	Text result
ATTRIBUTE(*screen-name*,*field-name*)	(10,20),3,BRI,NUM
ATTRIBUTE(*screen-name*,(1,40))	(1,35),HED,BRI
ATTRIBUTE(TERMINAL)	(24,80),BRI,REV,BLI,UND
ATTRIBUTE(TERMINAL,CURSOR)	(2,24)
ATTRIBUTE(PRINTER)	(60,132),BRI,UND

BIG

Use the BIG statement to define numeric variables or arrays of numeric variables that allocate space in the work area to hold their value with a maximum of 16 significant digits. Values are initially set to zero.

BIG *big-name* [(*dimension*,...)],...

big-name

Description	*Required*. Define the symbolic name of the numeric variable or array.
Format	A MANTIS symbolic name
Consideration	No processing occurs if *big-name* is already defined.

dimension

Description	*Optional*. Specify an array dimension.
Format	Arithmetic expression that evaluates to a value in the range 1–*max*, where *max* is the value of the MAXDIMSIZE MANTIS Option.

Considerations

- ♦ Use one dimension parameter to specify a one-dimensional array, two dimension parameters to specify a two-dimensional array, and so on.
- ♦ The maximum number of dimensions you can specify is determined by the value of the MAXNODIMS MANTIS Option. Each dimension specifies the maximum value of the corresponding array subscript.

General considerations

- The maximum number of variable names you can specify is determined by the value of the MAXVARS MANTIS Option.
- Binary operations involving mixed operand data types cause one operand to be converted to a compatible data type. The coercion rules are discussed in "Numeric considerations" on page 51.
- Auto mapping of variables takes place during complex variable declarations such as FILE and ACCESS statements. MANTIS generates a dissimilarity fault (212) if you attempt to map a new variable to an existing one with less precision. For example, if an ACCESS statement would define a DECIMAL variable with a precision of 10, and that variable is already defined in the work area with a precision of less than 10, a 212 fault will occur. For the purposes of dissimilarity fault checking, MANTIS data types are attributed with the following precision.

Type	Precision
SMALL	6
INTEGER	9
BIG	14
DECIMAL	1–31 as defined

- So for example, it is invalid to auto-map a BIG onto an existing SMALL or INTEGER variable, and a DECIMAL(15) cannot be mapped onto an existing BIG variable.
- See also "SMALL" on page 344, "DECIMAL" on page 175, and "INTEGER" on page 264.

Example

```
10 X=15
20 BIG ALPHA(64,3),BETA(X)
```

BREAK

Use the BREAK statement to exit from a FOR-END, UNTIL-END, WHEN-END, or WHILE-END statement. The statement after the END statement is executed next.

BREAK

General considerations

- With nested logic statements, BREAK terminates execution of the innermost FOR-END, UNTIL-END, WHEN-END, or WHILE-END block of statements where it occurs.
- BREAK from a WHEN-END block of statements bypasses any other WHEN conditions.
- See also “NEXT” on page 286 and “RETURN” on page 320.

Examples

```
 10 FOR L=1 TO MAXLINES:|                   For each screen line
 20 .GET CUSTOMER LEVEL=L:|                 Get customer detail fields
 30 .IF CUSTOMER="END":|                    Check status from GET
 40 ..BREAK:|                               Exit FOR Loop if end of file
 50 .END
 60 END
 70 CONVERSE CUST_DETAILS:|                 Display customer details
110 WHEN CODE="R"
120 .COLOR="RED"
130 .BREAK
140 WHEN CODE="B"
150 .COLOR="BLUE"
160 .BREAK
170 WHEN CODE="G"
180 .COLOR="GREEN"
190 .BREAK
200 END
```

CALL

Use the CALL statement to call an interface program. MANTIS calls the program specified in the interface design and makes the status returned by the program available.

CALL *interface-name* [(*argument*,...)][LEVEL=*level-number*]

interface-name

Description *Required*. Specify the symbolic name of an interface.

Format MANTIS symbolic name as defined in a previously executed INTERFACE statement.

argument

Description *Optional*. Provide an argument to be passed to the corresponding element in the interface area definition (the value of the first argument is passed to the first element in the interface area definition; the value of the second argument is passed to the second element, and so on.)

Format Arithmetic or text expression

Considerations

- MANTIS stores the values of the arguments you supply in the corresponding MANTIS variables specified in the interface profile. Before calling the interface program, all of the MANTIS variables in the interface profile are converted to the specified format and stored in the interface program's work area. When the interface program exits, the values in the work area are converted back and stored in the MANTIS variables.

- When the argument that is passed to an interface is an array, each element of the array must be passed separately.

LEVEL=*level-number*

Description	*Optional*. Specify which level of MANTIS array elements should be passed to the interface program.
Default	LEVEL=1
Format	Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding INTERFACE statement.
Consideration	You must specify LEVEL=*level-number* if the corresponding INTERFACE statement specifies *levels* unless you want the default value.

General considerations

- ♦ The eight-character text value at the start of the interface program's work area is set to spaces before the interface program is called and is returned to MANTIS when the program exits. To obtain the status value, use the "interface-name" built-in text function.
- ♦ Use only after discussing with the Master User the interfaces that are available to you.
- ♦ You can use CALL to build menu-driven systems where the menu itself and some system components are written in MANTIS. From MANTIS you can use the CALL statement to invoke old or performance-sensitive components written in COBOL, BASIC, FORTRAN, MACRO Assembler, PL/1, and other languages.

If the interface program writes to the screen it may interfere with optimized screen refreshing performed by subsequent MANTIS screen I/O statements. You have two ways of restoring the MANTIS screen appearance after a CALL statement:

- Use the CLEAR statement in your program, following the CALL statement. You can decide if CLEAR is needed based on your knowledge of how specific interface programs behave.
- Enable the AUTOREFRESH MANTIS Option, which unconditionally refreshes the full screen on the next screen I/O, regardless of whether the interface program wrote to the screen.

In either case the screen is not restored until the next screen I/O statement. CLEAR has a more immediate effect than AUTOREFRESH when the screen is operating in scroll mode and in full screen mode CLEAR erases the entire mapset whereas AUTOREFRESH has no effect on the mapset.

Example

```
20 INTERFACE MASTER("CUSTOMERS","ALIBABA",10)
30
40
50
60 CALL MASTER("GET",1234) LEVEL=2
```

CHAIN

Use the CHAIN statement to replace the program currently in the work area or at the current DOLEVEL with another MANTIS program and begin executing that program. MANTIS terminates the issuing program and erases all variables after evaluating any arguments being passed.

CHAIN *libname*[,*argument*,...][LEVEL]

libname

Description *Required*. Specify the library name of the program you want to load and execute.

Format Text expression that evaluates to a library name in the format:

```
[user-name:]program-name
```

Consideration If the program is in your own library, you do not need to specify *user-name*.

argument

Description *Optional*. Specify the argument(s) to be passed to the new program.

Format Expressions or symbolic names of variables or arrays

Considerations

- ♦ The arguments may not be symbolic SCREEN, PROGRAM, ENTRY, INTERFACE, FILE, ACCESS, ULTRA, or RDM VIEW names.
- ♦ If you are operating in IBM compatibility mode, the arguments must correspond in type and number with the ones specified in the ENTRY statement of the program that was chained to. Otherwise, if you supply fewer arguments than necessary, MANTIS supplies extras and sets them to zero.
- ♦ The arguments in a CHAIN statement define the type and dimensions for the corresponding variables in the program that you are chaining to.

LEVEL

Description *Optional*. Indicate that only the program at the current DOLEVEL should be overlaid.

Considerations

- MANTIS replaces the program at the current DOLEVEL and begins executing the CHAIN program.
- CHAIN LEVEL preserves address ability to any reference parameter as long as the corresponding formal argument is passed in the CHAIN argument list. For example:

```
ENTRY SUB(FORMAL)
.CHAIN "SUB_A",FORMAL LEVEL
EXIT
```

  As long as the FORMAL argument is present in CHAIN LEVEL argument lists, address ability to the actual parameter in the original caller of SUB is maintained.

General considerations

- If your program will execute a CHAIN statement, save the program before you run it in programming mode; otherwise, any changes you have made will be lost when it is replaced by the chained program.
- You can chain to any program in any user's library. When the program stops, however, you will not be allowed to LIST or modify it unless its password is the same as the password of the program it replaced in the work area.
- You can pass data from one program to another by using the CHAIN statement. This requires an ENTRY-EXIT statement in the program to which you are chaining.
- The maximum number of arguments allowed is determined by the value of the MAXCHARGS MANTIS Option. To maintain compatibility with MANTIS for the IBM mainframe, this is set to 40 when IBM compatibility mode is enabled.

- See "Programming techniques" on page 415 for additional information on the CHAIN statement as it is used with external DO.
- Any records HELD by doing an ENQUEUE or GET with ENQUEUE are released by the CHAIN statement if the RELCHAIN MANTIS Option is set to TRUE.
- If the COMCHAIN MANTIS Option is set, then a COMMIT will be performed each time the CHAIN statement without LEVELs is executed.
- If the NOCOMCHAIN MANTIS Option is set, then MANTIS is prevented from performing a COMMIT each time the CHAIN statement without LEVELs is executed.
- If the RELCHAIN MANTIS Option is set, then a RELEASE will be performed each time the CHAIN statement without LEVELs is executed.
- If the NORELCHAIN MANTIS Option is set, then MANTIS is prevented from performing a RELEASE each time the CHAIN statement without LEVELs is executed.
- The COMCHAIN and RELCHAIN MANTIS Options must be set for IBM compatibility.
- Executing a STOP or the main ENTRY's EXIT statement will cause an automatic CHAIN to the user's defined Facility program when run outside programming mode. For details about the facility program specification, refer to *AD/Advantage MANTIS Administration OpenVMS/UNIX*, P39-1320.
- See also "STOP" on page 349, "PROGRAM" on page 308, "DO" on page 199, and "ENTRY-EXIT" on page 206.

Example

```
 20.HEAD"FACILITY SELECTION"
 30 .CLEAR
 40 .SHOW"DESIGN A PROGRAM ...... 1"
 50 .SHOW"DESIGN A SCREEN ....... 2"
 60 .SHOW"TERMINATE ............. CANCEL"
 70 .OBTAIN OPTION
 80 .WHEN KEY="PF1"OR OPTION=1
 90 ..CHAIN"CONTROL:PROGRAM-DESIGN"
100 .WHEN KEY="PF2" OR OPTION=2
110 ..CHAIN"CONTROL:SCREEN-DESIGN"
120 .WHEN KEY="CANCEL"
132 ..CHAIN "MASTER:TERMINATE"
140 .END
```

CHR

Use the CHR function to return a text value consisting of the character(s) with ASCII code(s) “a”,....

CHR(*a*,...)

a

Description *Required*. Specifies the arithmetic expression whose characters you want returned.

Format Arithmetic expression in the range 0–255

General considerations

- MANTIS ignores fractions. For example, CHR(97,97.1,97.9) returns aaa.
- If the value of the expression is outside the valid range, MANTIS attempts to use the value MOD 256. For example, CHR(97,2**22+97, -(2**22)+97) returns aaa.

Example

```
SHOW CHR(97,98,99)
abc
```

CLEAR

Use the CLEAR statement to clear the screen, clear the data referred to by a symbolic name, clear all program data, or clear a fault trap.

```
        ┌ name   [,...] ┐
CLEAR   │ ALL           │
        └ TRAP          ┘
```

name

Description *Optional*. Specifies a symbolic name as defined in an ACCESS, BIG, DECIMAL, FILE, INTEGER, INTERFACE, SCREEN, SMALL, TEXT, ULTRA, or VIEW statement. MANTIS clears the data referred to by the specified symbolic name.

ALL

Description *Optional*. Clears the data referred to by all symbolic names in your program.

Consideration MANTIS clears the values of all variables and entities defined in the current program. Data in programs or subroutines at higher or lower levels is not affected.

TRAP

Description *Optional*. Clears a fault trap set by SET TRAP.

Consideration MANTIS clears all TRAPs at this DOLEVEL. TRAPs at higher DOLEVELs are still in effect.

General considerations

- ♦ CLEAR with no parameter

  Clears the Map Display (except the heading in scroll mode) by displaying the required number of blank lines and also clears the map set. Program variables are not affected. When CLEAR is used as a command, the screen is cleared immediately. When CLEAR is used as a statement, the screen is cleared just before output from a CONVERSE, SHOW or OBTAIN statement is displayed.

- ♦ CLEAR with the symbolic name of a INTEGER, SMALL, BIG or DECIMAL variable or array

  MANTIS sets the value of the variable (or each element of the array) to zero.

- ♦ CLEAR with the symbolic name of a TEXT variable or array

  MANTIS sets the current length of the variable (or each element of the array) to zero, giving it the null value “”.

- ♦ CLEAR with the symbolic name of an entity defined by an ACCESS, FILE, INTERFACE, SCREEN, ULTRA, or VIEW statement.

  MANTIS clears the values of all variables and arrays associated with the entity. In addition, MANTIS clears the value that is returned when the symbolic name is used to invoke a built-in text function.

- ♦ See also “RESET MAP MODIFIED” on page 129.

Example

```
10 BIG COUNT,SUBTOTAL(10)
20 FILE CUSTFILE("CUSTOMERS","XANADU")
30 SCREEN CUSTSCREEN("CUSTOMER")
40 ...
50 ...
60 CLEAR COUNT,SUBTOTAL,CUSTFILE,CUSTSCREEN
```

COMMIT

Use the COMMIT statement to indicate the completion of a Logical Unit of Work (LUW). Updates on ULTRA file and SUPRA RDM views are committed to the SUPRA database and CIS_TM files, preventing them from being backed out by RESET or system failure. Updated buffers for the MANTIS file and external files are flushed. Locked records are unlocked. All delayed releases of external files are performed.

NOTE

All references to RU Journaling apply to OpenVMS only. All references to RDM and ULTRA apply to SUPRA RDM users only.

```
COMMIT  [ON ]
        [OFF]
```

ON

Description *Optional*. Indicates that you want MANTIS to perform COMMIT processing before each read operation on your terminal (for example, on every CONVERSE, OBTAIN, or WAIT statement).

Consideration MANTIS enables automatic COMMIT processing by default.

OFF

Description *Optional*. Indicates that you do not want MANTIS to perform COMMIT processing before each read operation on your terminal.

Consideration Automatic COMMIT processing is initially enabled. If the Logical Units of Work in your application are not synchronized with terminal reads, use COMMIT OFF to disable automatic COMMIT processing and use COMMIT by itself to indicate the completion of each Logical Unit of Work.

General considerations

- ♦ COMMIT with no parameter
  - MANTIS commits ULTRA and RDM updates to the SUPRA database, and RU Journal led files, respectively.
  - MANTIS flushes any updated buffers for the MANTIS file and for any external files that were opened by ACCESS statements.
  - MANTIS unlocks any records that were locked with GET...ENQUEUE.
  - MANTIS releases any resources that were held with an ENQUEUE statement.
  - All delayed releases of ACCESSes are performed and files are freed. See also “RELEASE (statement)” on page 315.
- ♦ If you are using RU Journaling, a COMMIT causes END Recovery Unit Processing.
- ♦ In addition, any external file that is opened (by an ACCESS statement) in an external subprogram is automatically closed by MANTIS when the subprogram exits; however, an RU journal led file cannot be closed when there are uncommitted updates. Therefore, MANTIS external subroutines which execute ACCESS statements should generally perform COMMIT or RESET prior to returning via EXIT or RETURN; otherwise, MANTIS may delay closing the file until the next implicit or explicit COMMIT or RESET. You should also keep in mind that the COMMIT or RESET affects all currently opened external files at all DOLEVELS.

Example

```
20 SCREEN MAP("CUSTOMER")
.
.
.
100 COMMIT
110 CONVERSE MAP
```

CONVERSE

Use the CONVERSE statement to display a map set consisting of one or more screens or to add a screen map to the map set without displaying it. Screen fields show the values of corresponding MANTIS variables. Unprotected screen fields can be updated from the keyboard. When control returns to MANTIS, the MANTIS variables are updated.

NOTE

See the discussion of the MANTIS Logical Terminal Interface in "Programming techniques" on page 415 for detailed examples of CONVERSE.

$$\textbf{CONVERSE}\ \textit{screen - name}\ \left[\begin{array}{l} [(\textit{row}1,\textit{col}1)] \left[\begin{array}{l}\textbf{WAIT}\\ \textbf{SET}\\ \textbf{UPDATE}\end{array}\right] \left[\left\{\begin{array}{l}\textbf{WINDOW}\\ \textbf{DISPLAY}\end{array}\right\}\right] \\ \\ [(\textit{row}2,\textit{col}2)]\,[\textit{template}]\,\textbf{[TIME =}\ \textit{timeout}\,\textbf{]} \\ \textbf{RELEASE}\end{array}\right]$$

screen-name

Description *Required*. Specifies the symbolic name of a screen.

Format MANTIS symbolic name as defined in a previously executed SCREEN statement

Consideration Use CONVERSE *screen-name* to display the specified screen by itself. The screen is the only map in the map set. Both the map domain and the window are positioned at row 1, column 1 of the logical display, so the screen looks the same as it did in Screen Design.

(*row1*,*col1*)

Description *Optional*. Specifies the position in the logical display of the top left corner of the map domain.

Format *row1* must be an arithmetic expression that evaluates to a row number in the logical display. Rows are numbered 1 to 32767 from top to bottom. *col1* must be an arithmetic expression that evaluates to a column number in the logical display. Columns are numbered 1 to 32767 from left to right.

Considerations

- ♦ The row and column values may be negative when you are specifying the position of a floating map in the logical display. This allows the floating map to be located relative to the bottom row (*row1* negative) or the right-hand edge (*col1* negative) of the physical screen.
- ♦ You can only use negative values if the corresponding windowing attribute of the map is disabled. To specify your screen design as a floating map, refer to the vertical and horizontal windowing map-level attributes in *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.
- ♦ All row/column coordinates are relative to row 1, column 1. Use (*row1*,*col1*) and (*row2*,*col2*) if you want to position the screen or the window at a different position to the 1,1 default.

WAIT

Description *Optional*. Indicates that the specified map should be added to the map set without performing an I/O.

Considerations

- ♦ All other maps in the map set become passive maps.
- ♦ The contents of the window are not displayed on the screen.
- ♦ CONVERSE...WAIT adds the specified map to the map set, optionally specifying its (*row1*,*col1*) position. Other parameters are ignored.

SET

Description *Optional*. Indicates that the specified map should become the active map in the map set.

Considerations

- ♦ All other maps in the map set become passive maps. The contents of the window are displayed on the screen.
- ♦ Update of fields in passive maps is not allowed.
- ♦ CONVERSE...SET or CONVERSE...UPDATE adds the final map to the current map set and displays the contents of the window.

UPDATE

Description *Optional*. Indicates that the specified map should become the active map in the map set.

Considerations

- ♦ All other maps in the map set become passive maps. The contents of the window are displayed on the screen.
- ♦ Update of fields in passive maps is allowed.
- ♦ CONVERSE...SET or CONVERSE...UPDATE adds the final map to the current map set and displays the contents of the window.

WINDOW

Description *Optional*. Displays the position of the window, and specify (optional) where to put the window in the logical display using the (*row2*, *col2*) parameter.

Considerations

- ♦ The row and column coordinates of the top left corner of the window are displayed in a floating map which occupies the last twelve columns of the second last line of the screen.
- ♦ You can turn off this display using the WINDOWMAP (GOLD/W) terminal function.

DISPLAY

Description *Optional*. Specifies the position of the viewing window in the logical display, using (*row2*,*col2*).

(*row2*,*col2*)

Description *Optional*. Specifies the position in the logical display of the top left corner of the window with WINDOW or DISPLAY options.

Format *row2* must be an arithmetic expression that evaluates to a row number in the logical display. Rows are numbered 1 to 32767, from top to bottom. *col2* must be an arithmetic expression that evaluates to a column number in the logical display. Columns are numbered 1 to 32767, from left to right.

template

Description *Optional*. Specifies the name of a TEXT variable containing additional screen heading information that is displayed in the window.

Considerations

- ♦ Note that this feature is non-portable and terminal dependent.
- ♦ The template parameter can be used to display character graphics not available through the Screen Design Facility. Assign the correct escape sequences and associated data to the template text variable.
- ♦ MANTIS returns the cursor to home before the template is written.
- ♦ Do not change any video attributes or MANTIS may lose track of them.
- ♦ ATTRIBUTE(*screen-name*) = “ALARM” rings the terminal bell before a CONVERSE.
- ♦ **OpenVMS** Under OpenVMS the runtime library routines for screen management, SMG, are used to optimize refreshing of screens. SMG does not allow inclusion of escape sequences. Therefore, templates containing escape sequences must turn OFF use of SMG. See the considerations for the ATTRIBUTE(TERMINAL)= statement.

timeout

Description *Optional*. Specifies either a timeout period for the input phase, or no input (that is, a “display only” CONVERSE) with a delay period.

Format A numeric expression

Default None; no timeout period applies.

Considerations

- ♦ A positive timeout value (integer portion only) specifies the number of seconds to wait, after the last keystroke is made, before the input phase of the CONVERSE is terminated.
- ♦ A zero or negative timeout value specifies that the CONVERSE be performed with no input at all. In this case the absolute value of timeout specifies the number of seconds to delay after the CONVERSE is performed.
- ♦ When a CONVERSE is terminated due to timeout (including timeout <= 0), the value of the screen-name built-in returned to the MANTIS program is “TIMOUT.”
- ♦ The TIME parameter is not allowed if the WAIT parameter is used.

RELEASE

Description *Optional*. Removes the specified map from the map set. The resulting map set is not displayed until the next CONVERSE statement.

General considerations

- ♦ To create a map set with more than one map, use one or more CONVERSE statements with the WAIT parameter, followed by one CONVERSE statement with either the SET or the UPDATE parameter.
- ♦ When you issue a CONVERSE statement, MANTIS:
  - - Determines which fields in each map are visible in the window, taking into account the front-to-back order of the map set. Some fields may be overlaid by fields in maps which are in front of them.

- Sets the contents of each data field to the value of the corresponding variable. Numeric values are formatted using the edit mask for the field if it has one; otherwise, the standard decimal format is used.

- Determines which data fields should be protected to prevent them from being updated. A field is protected if it is partially overlaid by another field or it is partially outside the window, unless it is in a map which has the automatic windowing attribute. All fields in a passive map are protected unless CONVERSE...UPDATE is used.

- When conversing maps using the UPDATE option of the CONVERSE statement, the following rule governs whether you can update partially displayed fields on a passive map. Fields on a passive map that are partially displayed because they are overlaid by the active map cannot be updated; fields on a passive map that are partially displayed because they overlap the physical screen boundary can be updated if the active map has automatic windowing.

- Sets up a message field using any data which is pending from earlier SHOW statements (for example, the preceding SHOW statements terminated by a semicolon or comma, also called unterminated SHOW statements). The field chosen is either a visible MESSAGE field or (if no such field exists) the Message Field of the converse input map.

- Displays the contents of the window on the screen and waits for your response.

♦ You respond (using the keyboard) and:

  - Use the data-entry keys to update unprotected data fields on the screen. You can also enter data in the Unsolicited Input Field and the Reply Field on the bottom line of the screen.

  - Use terminal functions to perform actions such as scrolling. MANTIS then repeats the above steps to display the new contents of the window. Control does not return to the MANTIS program.

  - Press RETURN or enter a program function sequence to return control to the MANTIS program.

Before returning control to the program, MANTIS:

- Uses the contents of each updated data field to update the value of the corresponding MANTIS variable. Numeric fields are converted using the edit mask for the field if it has one; otherwise, the standard decimal conversion is performed. MANTIS performs any extended edit validation (such as range checking) that you specified in Screen Design. If numeric conversion or validation fails, MANTIS displays an error message on the second last line and highlights the fields to be corrected.
- Saves a text value identifying the program function that you entered (or the value that you entered in the Reply Field). This value can be obtained by using the symbolic name of the active map to invoke a built-in text function (see also “screen-name” on page 324).
- Saves any data you entered in the Unsolicited Input Field on the bottom line of the screen. This data can be obtained by the first OBTAIN statement that is executed after the CONVERSE and before any SHOW statement. For example: CONVERSE MAP:OBTAIN REPOINT. The OBTAIN statement reads from the saved unsolicited input field.

♦ Overlapping fields have precedence of order when they are CONVERSEd. Later fields have priority over earlier fields.

♦ You can display a map only once within a map set. If you add it twice (with a WAIT, SET, or UPDATE option), it moves to the new dynamic offset. If you want the same screen image to appear twice, you must have two screen variables and converse both of them, for example :

```
SCREEN MAP1("INDEX"),MAP2("INDEX")
CONVERSE MAP1
CONVERSE MAP2(20,30)SET
```

♦ If a map set (or part of a map set) is conversed directly via external DO, MANTIS clears the entire map set when it exits from the external DO. To avoid clearing the portion of the map set conversed in the calling program, release the maps defined in the external DO using the CONVERSE map RELEASE statement.

- ♦ If you converse a screen that contains double height/width fields on a device that does not support this feature (or if the feature is disabled), MANTIS displays all fields with single height, single width format. No warnings are issued. This also occurs in IBM compatibility mode.

- ♦ If you disable double height for a device, double height fields are displayed as single height, double width. If you disable double width for a device, double height is implicitly disabled as well. As a result, fields with double height or double width attributes are displayed with single height, single width.

- ♦ If a screen with a double width field is conversed over another screen, and the double width fields overlay single width fields, the single width line is automatically switched to a double width. Similar actions occur if a double height field overlays a single height field from a previously conversed screen. No warnings are issued.

- ♦ **OpenVMS** Scroll-mode output is always performed with the terminal in normal mode (for example, no color). When the terminal is full-screen mode, normal mode is used as long as there are no color fields in the logical display (whether in view or not). When a screen with color fields is conversed in the logical display, the terminal switches to ReGIS mode (if possible) and remains in that mode as long as there are screens in the logical display with color fields.

- ♦ See also “OBTAIN” on page 290, “WAIT” on page 398, “SHOW” on page 334, “PROMPT” on page 311, and “SCREEN” on page 322.

Examples

```
10 FILE RECORD("INDEX","SERENDIPITY")
20 SCREEN MAP("INDEX")
30 UNTIL RECORD="END" OR MAP="CANCEL"
40 .GET RECORD
50 .CONVERSE MAP
60 END
```

```
10 SCREEN CUSTOMER("CUSTOMER")
20 SCREEN SALESMAN("SALESMAN")
30 CONVERSE CUSTOMER WAIT
40 CONVERSE SALESMAN (10,1) UPDATE
80 CONVERSE SALESMAN RELEASE
```

COS

Use the COS function to return the cosine of an angle where the angle is expressed in radians.

COS(*a*)

a

Description *Required*. Specifies the angle (in radians) whose cosine you want returned.

Format Arithmetic expression

Considerations

- The expression result is coerced to BIG. Decimal expression results may therefore lose some precision or may generate an arithmetic fault (310) if the magnitude is too large for a BIG data type.
- See also “ATN” on page 110, “PI” on page 303, “SIN” on page 338, and “TAN” on page 352.

Example

```
100 X=10
110 Y=COS(X)
120 SHOW X,Y
RUN
10          -0.8390715291
```

CURSOR

Use the CURSOR function to determine the location of the cursor during the last terminal input.

$$\textbf{CURSOR}\left\{\begin{array}{l} \mathbf{(}\textit{screen - name}\mathbf{,}\left\{\begin{array}{l}\textit{field - name}\\ \mathbf{(}\textit{lrow},\textit{lcol}\mathbf{)}\end{array}\right\}\ ) \\ \\ \textbf{("FIELD")} \\ \textbf{("SCREEN")} \\ \textbf{("INDEX1")} \\ \textbf{("INDEX2")} \end{array}\right\}$$

screen-name, *field-name*

Description Use this form of the CURSOR function to test whether the cursor appeared in a specific field at the last terminal input. Specify the symbolic name of the screen and field you are testing. MANTIS returns TRUE if the cursor appeared within the domain of the field you specified when a function key was pressed. Otherwise, MANTIS returns FALSE.

Format *screen-name* is a MANTIS symbolic name as specified in a previously executed SCREEN statement. *field-name* is a MANTIS symbolic name of a field as specified in Screen Design. *field-name* may be a subscripted repeat field.

screen-name, (l*row*,l*col*)

Description Use this form of the CURSOR function to test whether the cursor appeared in a specific field at the last terminal input. Specify the row and column logical display coordinates of the field you are testing. The selected field is the one in the specified screen that contains these coordinates. MANTIS returns TRUE if the cursor appeared within the domain of the field you specified when a function key was pressed. Otherwise, MANTIS returns FALSE.

Format *screen-name* is a MANTIS symbolic name as specified in a previously executed SCREEN statement. *lrow* and *lcol* must each be a numeric expression that evaluates to a value from 1 to 32767.

"FIELD"

Description	Specifies that MANTIS should return the text string equivalent of the symbolic name of the field in which the cursor appeared at the last terminal input.
Consideration	If the cursor was not in any field at the last terminal input, a zero length text string is returned.

"SCREEN"

Description	Specifies that MANTIS should return the text string equivalent of the symbolic name of the screen containing the field in which the cursor appeared at the last terminal input.
Consideration	If the cursor was not in any field at the last terminal input, a zero length text string is returned.

"INDEX1"

Description	Specifies that MANTIS should return the index of the first dimension of the repeat field in which the cursor appeared at the last terminal input.
Consideration	If the cursor was not in a repeat field at the last terminal input, zero is returned.

"INDEX2"

Description	Specifies that MANTIS should return the index of the second dimension of the repeat field in which the cursor appeared at the last terminal input.
Consideration	If the cursor was not in a two-dimensional repeat field at the last terminal input, zero is returned.

General considerations

- MANTIS returns TRUE if the cursor appeared within the domain of the field you specified when a function key was pressed. Otherwise, MANTIS returns FALSE.
- MANTIS returns FALSE when the map is not in the current map set.
- See also “ATTRIBUTE (TERMINAL,CURSOR)” on page 115.

Examples

Select items from a list of options by cursor positioning. For example, select an item from a list of choices (menu processing) or process an element from a list:

```
100 WHEN CURSOR(MENU_SCREEN,CLIENT_UPDATE)
110 .DO CLIENT_PROCESSING
120 WHEN CURSOR(MENU_SCREEN,INVENTORY_UPDATE)
130 .DO INVENTORY_PROCESSING
140 .
150 .
160 .
300 END
```

Position the cursor to scroll to a group of lines. For example, MANTIS scrolls up or down when CURSOR is the scroll amount.

```
 10 WHILE I<=COUNT
 20 .IF CURSOR(CLIENT_SCREEN,NAME(I))
 30 ..FIRST_KEY=NAME(I)
 40 ..I=COUNT:|END LOOP
 50 .END
 60 I=I+1
 70 END
 80 GET REC(FIRST_KEY)LEVEL=1
 90 I=1
100 WHILE I<COUNT AND REC<>"END"
110 .I=I+1
120 .GET REC LEVEL=I
130 END
```

Issue a help prompt based on the cursor location when the HELP function key sequence is invoked.

```
 10 SCREEN MAP("CLIENT_SCREEN")
 20 TEXT HELPFLD
 30 CONVERSE MAP
 40 HELPFLD=CURSOR("FIELD")
 50 IF MAP="HELP" AND HELPFLD<>NULL
 60 .PROMPT HELPFLD
 70 END
```

DATAFREE

Use the DATAFREE function to return the number of bytes remaining in your Data Work Area.

DATAFREE

General consideration

- This function is not relevant in the MANTIS environment, but does provide for compatibility with MANTIS for the IBM mainframe.

Example

```
SHOW DATAFREE
32749
```

DATE (function)

Use the DATE function to return a text string containing the current date. The format of the date is determined by a previous date mask specification using the DATE statement.

DATE

General considerations

- The current date mask setting is inherited on an external DO or CHAIN LEVEL, and is restored upon subprogram EXIT. This inheritance does not occur in the special case of when a non-privileged user's program attempts to externally DO or CHAIN LEVEL a privileged user's program.
- For non-privileged users' programs, if the date mask has not been previously specified, or has been set to NULL (""), the DATE function returns the current date in the format specified by the DATEMASK MANTIS Option. If the DATEMASK MANTIS Option value is also NULL ("") then YY/MM/DD format is used.
- A CHAIN (without LEVEL) will reset the date mask to NULL ("").
- See also "DATE (statement)" on page 173, "TIME (function)" on page 356, and "TIME (statement)" on page 357.

Example

```
 30 ..TEXT LAST_UPDATED(20)
 40 ..DATE="YYYY/MM/DD"
 50 ..WHEN MAP="ENTER"
 60 ...ERROR=FALSE
 70 ...DO EDIT_FIELDS
 80 ...IF NOT(ERROR)
 90 ....LAST_UPDATED=DATE
100 ....UPDATE REC
110 ...END
```

DATE (statement)

Use the DATE statement to specify the date mask to be used by the DATE function.

DATE=*mask-expression*

mask_expression

Description *Required*. Specify the date mask to be used by the DATE function.

Format Text expression of 0–255 characters.

Considerations

- If the mask specified is longer than 255 characters, the first 255 characters are used.
- The following special strings are substituted with the data described when the DATE function is invoked. Any characters in the mask that are not part of one of these special strings are treated as punctuation and are not translated.

DDD or ddd	day of year (Julian day 001–366)
DD or dd	day of month (01–31)
MMM or mmm	month name (JAN–DEC)
MM or mm	month (01–12)
YYYY or yyyy	4-digit year (0000–9999)
YY or yy	2-digit year (00–99)

- Where there are ambiguities on substitution, the substitution takes place in the order of the special strings listed above.
- There is a special mask escape character (%). Any character following this escape is taken literally and the escape can be used to break up what might otherwise be a valid substitution string.
- When alphabetic substitution is required (for example, MMM), the case of any special string characters are reproduced on substitution. Case is not important for numeric substitution.

General consideration

- See also “DATE (statement)” on page 173, “TIME (function)” on page 356, and “TIME (statement)” on page 357.

Examples

```
DATE="DD-Mmm-YYYY"
SHOW DATE
18-Aug-1992

DATE="Summer of 'YY"
SHOW DATE
Su08er of '92

DATE="Sum%mer of 'YY"
SHOW DATE
Summer of '92
```

DECIMAL

Use the DECIMAL statement to define numeric variables or arrays of numeric variables that allocate space in the work area to hold up to 31 significant digits. Values are initially set to zero.

DECIMAL numbers are stored internally as ASCII digits and are therefore the most accurate way of storing decimal numbers with fractions. Arithmetic operations on DECIMAL numbers are done in software and are very slow compared with BIG/SMALL/INTEGER arithmetic. The domain of DECIMAL is approximately +/-9.9D-128 to +/-9.9D127. The greater the precision specified, the more decimal places allowed after the decimal point. D-notation is discussed in "Numeric conversion" on page 51.

DECIMAL [(*precision*)] *decimal-name* [(*dimension*,...)],...

precision

Description	*Optional*.. Specify the precision of the decimal number or array elements. The precision is the maximum number of significant digits that can be stored.
Default	18
Format	Arithmetic expression in the range 1–31
Consideration	On assignment to a DECIMAL variable, the precision of the target may be less than that of the value being assigned, then the value is rounded to the precision of the target variable or array element.

decimal-name

Description	*Required*. Define the symbolic name of the numeric variable or array.
Format	A MANTIS symbolic name
Consideration	No processing occurs if *decimal-name* is already defined.

dimension

Description *Optional*. Specify an array dimension.

Format Arithmetic expression that evaluates to a value in the range 1–*max*, where *max* is the value of the MAXDIMSIZE MANTIS Option.

Considerations

- Use one dimension parameter to specify a one-dimensional array, two dimension parameters to specify a two-dimensional array, and so on.
- The maximum number of dimensions you can specify is determined by the value of the MAXNODIMS MANTIS Option. Each dimension specifies the maximum value of the corresponding array subscript.

General considerations

- The maximum number of variable names you can specify is determined by the value of the MAXVARS MANTIS Option.
- All binary operations involving only one DECIMAL operand cause the other operand to be converted to DECIMAL. Coercions of this nature are done to 31 digits of precision so that no information is lost.
- All DECIMAL arithmetic and function results have 31 digits of precision available so that complex calculations tend not to lose accuracy. Rounding to less than 31 significant digits will occur on assignment to a DECIMAL variable or array element whose defined precision is less than that of the intermediate expression result.

- Auto-mapping of variables takes place during complex variable declarations such as FILE and ACCESS statements. MANTIS generates a dissimilarity fault (212) if you attempt to map a new variable to an existing one with less precision. For example, if an ACCESS statement would define a DECIMAL variable with a precision of 10, and that variable is already defined in the work area with a precision of less than 10, a 212 fault will occur. For the purposes of dissimilarity fault checking, MANTIS data types are attributed with the following precision.

Type	**Precision**
SMALL	6
INTEGER	9
BIG	14

- For example, it is illegal to auto-map an INTEGER to an existing SMALL variable, and BIG fields cannot be mapped onto existing DECIMAL variables with less than 14 digits of precision.

- DECIMAL parameters to the mathematical functions ATN, COS, EXP, LOG, RND, SIN, SQR and TAN are coerced to BIG, possibly resulting in loss of precision, or an arithmetic fault (310), if the magnitude is too large for a BIG.

- DECIMAL operands of the exponentiation operator (**) are also coerced to BIG with the same risks engendered by the mathematical functions.

- BIG to DECIMAL conversion can lose accuracy which is not normally visible and is due to the fact that internal binary floating point format cannot always accurately represent decimal values. The following example illustrates this point.

```
BIG B
DECIMAL(31)D
B=PI/2
D=B:| Loss of accuracy may occur here
SHOW B=D,COS(B)=COS(D)
1            0
```

- MANTIS for the IBM mainframe imposes a restriction on DECIMAL multiplication and division operations. In DECIMAL multiplication, at least one of the two operands must have fewer than 16 significant digits. In DECIMAL division the divisor must have fewer than 16 significant digits. The precision may exceed these restrictions, it is the current value of the variables that is taken into consideration.

- See also “BIG” on page 144, “SMALL” on page 344, and “INTEGER” on page 264.

Examples Declare DECIMAL variables, scalar and array, with a precision of 31

```
10 DECIMAL(31) YEAR_REVENUE,MONTH_REVENUE(12)
```

Declare a scalar DECIMAL variable with a precision of 18

```
10 DECIMAL PROFIT
```

DELETE

Use the DELETE statement to delete a record from a MANTIS file, an external file, an ULTRA file, or a row from an RDM user view. Before you delete a record or row, you must first open it by processing an associated FILE, ACCESS, ULTRA, or VIEW statement. You do not need to read a record from a MANTIS file, an external file, or an ULTRA file before deleting it. You must, however, read the row from the view before deleting it.

DELETE (MANTIS file)

$$\textbf{DELETE}\ \textit{file-name}\begin{bmatrix}(\textit{key},\ldots)\ \ \textbf{ALL} \\ \textbf{ALL} \\ \textbf{LEVEL} = \textit{level-number}\end{bmatrix}$$

file-name

Description *Required*. Specifies the symbolic name of a MANTIS file from which you want to delete a record.

Format MANTIS symbolic name as defined in a previously executed FILE statement

key

Description *Optional*. Specifies the generic key value(s). All records beginning with the specified key value(s) will be deleted.

Considerations

- ♦ The data types of the supplied generic keys (*key*,...) must match the data types (text or numeric) of the corresponding elements in the file definition.
- ♦ If a numeric (INTEGER, SMALL, BIG or DECIMAL) key is supplied, it must match exactly in the corresponding positions of the record.
- ♦ Supplied TEXT keys are either truncated or space padded to the length of the corresponding FILE element. However, if the last (or only) key supplied is TEXT, no padding takes place.
- ♦ The total length of the generic key is the sum of the lengths of the corresponding file elements, with one exception—if the last supplied key is TEXT, the current length of the variable is used. For example, if a file has two TEXT keys each of length five, then

  ```
  DELETE F("ABC","DEF") ALL
  ```

  will delete records based on the generic key “ABCƀƀ DEF.”
- ♦ If any record is locked when you attempt to delete it using DELETE (*key*,...) ALL, the delete will terminate with a status of “HELD.”

ALL

Description *Optional*. Specifies that all records in the file are to be deleted..

Considerations

- ♦ To delete all records in the file, do not specify the key option.
- ♦ If any record is locked when you attempt to delete it using DELETE ALL, it is skipped and the next record is processed. If this happens, then the delete will give a “HELD” status after processing all records.

LEVEL=*level-number*

Description	*Optional*. Specifies which level of MANTIS array elements contains the key fields of the record you want to delete.
Default	LEVEL=1
Format	Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding FILE statement
Consideration	You must specify LEVEL=*level-number* if the corresponding FILE statement specifies *levels*, unless you want the default value (1) or you specify ALL.

General considerations

- Before you delete a record from a MANTIS file, you must first open it with a FILE statement. You do not need to read the record before deleting it.

- When ALL is not specified, the contents of the key fields of the FILE design identify the record to be deleted.

- You can obtain the status of the DELETE by using the “file-name“ built-in text function. If the DELETE was successful the FSI message field will indicate how many records were deleted. If the DELETE was unsuccessful, you can use the FSI message field or HELP LAST (in Program Design) to examine the system error message that may have been returned. This information is also available to MANTIS fault handling routines (see the SET TRAP statement). MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

  LOCK The password specified in the FILE statement for this file is not valid for deletions or insertions.

  ERROR A physical error occurred while deleting the record.

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

- ♦ **OpenVMS** If RMS Journaling is active and the file is RU Journal led, the following occurs:

  - The START_RU (Recovery Unit) occurs.
  - The record is locked until COMMIT or RESET.
  - You may back out the delete via RESET.

  During the START_RU, if RMS Journaling is active, all RU Journal led files opened for update (including the MANTIS File) automatically join the Recovery Unit. This implies that a release of any of these files is delayed until the next COMMIT or RESET.

- ♦ **OpenVMS** If RMS Journaling is not active, and the file is RU Journal led, MANTIS issues a fault.

Example

```
 10 ENTRY INDEX
 20 .FILE RECORD("INDEX","SERENDIPITY")
 30 .SCREEN MAP("INDEX")
 40 .GET RECORD
 50 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
 60 ..CONVERSE MAP
 70 ..WHEN MAP="PF1"
 80 ...INSERT RECORD
 90 ..WHEN MAP="PF2"
100 ...DELETE RECORD
110 ..WHEN MAP="PF3"
120 ...UPDATE RECORD
130 ..END
140 ..GET RECORD
150 .END
160 EXIT
```

DELETE (external file)

$$\textbf{DELETE}\ \textit{access}-\textit{name}\begin{bmatrix}(\textit{key},\ldots)\ \ \textbf{ALL}\\ \textbf{ALL}\\ \textbf{LEVEL}=\textit{level}-\textit{number}\end{bmatrix}$$

access-name

Description *Required*. Specifies the symbolic name of an external file from which you want to delete a record.

Format MANTIS symbolic name as defined in a previously executed ACCESS statement

Consideration If the file has been released, it is automatically reopened, but the position in the file is lost. For example, if a GET...NEXT is subsequently executed, the first record in the file is returned regardless of which record was being accessed before the release.

key

Description *Optional*. Specifies the generic key value(s). All records beginning with the specified key value(s) will be deleted.

Considerations

- ♦ The data types of the supplied generic keys (*key*,...) must match the data types (text or numeric) of the corresponding elements in the file definition.
- ♦ If a numeric (INTEGER, SMALL, BIG or DECIMAL) key is supplied, it must match exactly in the corresponding positions of the record.
- ♦ Supplied TEXT keys are either truncated or space padded to the length of the corresponding ACCESS element. However, if the last (or only) key supplied is TEXT, no padding takes place.

- The total length of the generic key is the sum of the lengths of the corresponding file elements, with one exception—if the last supplied key is TEXT, the current length of the variable is used. For example, if a file has two TEXT keys each of length five, then

  ```
  DELETE F("ABC","DEF") ALL
  ```

  will delete records based on the generic key “ABCƀƀ DEF.”

- If any record is locked when you attempt to delete it using DELETE (*key*,...) ALL, the delete will terminate with a status of “HELD.”

ALL

Description *Optional*. Specifies that all records in the file are to be deleted..

Considerations

- To delete all records in the file, do not specify the key option.
- If any record is locked when you attempt to delete it using DELETE ALL, it is skipped and the next record is processed. If this happens, then the delete will give a “HELD” status after processing all records.

LEVEL=*level-number*

Description *Optional*. Specifies which level of MANTIS array elements contains the key fields or the reference variable of the record you want to delete.

Default LEVEL=1

Format Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ACCESS statement

Consideration You must specify LEVEL=*level-number* if the corresponding ACCESS statement specifies *levels* unless you want the default value (1) or you specify ALL.

General considerations

- Before you delete a record from an external file, you must first open it with an ACCESS statement. You do not need to read the record before deleting it.
- For indexed files, when ALL is not specified, the contents of the key fields of the external file design identify the record to be deleted.
- You cannot delete records from sequential files.
- For relative files, the Relative Record Number (RRN) contained in the corresponding reference variable identifies the record to be deleted.
- You can obtain the status of the DELETE by using the “access-name“ built-in text function. If the DELETE was successful the FSI message field will indicate how many records were deleted. If the DELETE was unsuccessful, you can use the FSI message field or HELP LAST (in Program Design) to examine the system error message that may have been returned. This information is also available to MANTIS fault handling routines (see the SET TRAP statement). MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

  LOCK The password specified in the ACCESS statement for this file view is not valid for deletions or insertions.

  ERROR A physical error occurred while deleting the record or the RRN for relative files specified a record number that does not exist.

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

  OpenVMS If RMS Journaling is active and the file is RU Journal led, the following occurs:

  - The START_RU (Recovery Unit) occurs.
  - The record is locked until COMMIT or RESET.
  - You may back out the delete via RESET.

  During the START_RU, if RMS Journaling is active, all RU Journal led files opened for update (including the MANTIS File) automatically join the Recovery Unit. This implies that a release of any of these files is delayed until the next COMMIT or RESET.

- **OpenVMS** If RMS Journaling is not active, and the file is RU Journal led, MANTIS issues a fault.

Example

```
 10 ...
 20 .ACCESS RECORD("INDEX","SERENDIPITY",16)
 30 .SCREEN MAP("INDEX")
 40 .CONVERSE MAP
 50 .COUNTER=1
 60 .WHILE MAP<>"CANCEL" AND COUNTER < 17
 70 ..WHEN INDICATOR(COUNTER)="G"
 80 ...GET RECORD LEVEL=COUNTER
 90 ..WHEN INDICATOR(COUNTER)="D"
100 ...DELETE RECORD LEVEL=COUNTER
 .
 .
 .
 .
```

DELETE (ULTRA file)

NOTE

This statement applies to SUPRA PDM users only.

DELETE *ultra-name* [LEVEL=*level-number*]

ultra-name

Description	*Required*. Specifies the symbolic name of an ULTRA file from which you want to delete a record.
Format	MANTIS symbolic name as defined in a previously executed ULTRA statement

LEVEL=*level-number*

Description	*Optional*. Specifies which level of MANTIS array elements contains the key fields or the reference variable of the record you want to delete.
Default	LEVEL=1
Format	Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ULTRA statement
Consideration	You must specify LEVEL=*level-number* if the corresponding ULTRA statement specifies *levels* unless you want the default value (1).

General considerations

- ♦ Before you delete a record from an ULTRA file, you must first open it with an ULTRA statement. You do not need to read the record before deleting it.
- ♦ MANTIS obtains exclusive control of a record before requesting that ULTRA delete it. Therefore, you do not need to obtain the record using the ENQUEUE parameter on the GET statement.

- You can obtain the status of the DELETE by using the “ultra-name” built-in text function. The values are GOOD and HELD.

  GOOD MANTIS successfully deleted the record.

  HELD MANTIS failed to delete the record because it was held by another user. This situation normally calls for a finite number of retries of the GET statement, since individual records would not normally be held for long.

- MANTIS may also return a status of SETS, NOTFOUND, LOCK, or ERROR if TRAP is in effect.

  SETS You cannot delete a record from a primary file until you delete the associated chains from related files.

  NOTFOUND The chain set in a related file with the requested key does not exist.

  LOCK The password specified in the ULTRA statement is not valid.

  ERROR An error occurred while deleting the record.

  MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

Example

```
 30 SCREEN MAP(“DELETE_HISTORY”)
 40 ULTRA CUSTOMERS(“CLIENT”,”SALES”)
 50 ULTRA HISTORY(“PAYMENTS”,”TEXAS”,11)
 60 TEXT CUSTOMER_iD(20)
 70 SMALL LEVEL_NUMBER
 80 CUSTOMER_iD=”OUR-BEST”
 90 GET CUSTOMERS(CUSTOMER_iD)
100 LEVEL_NUMBER=1
110 GET HISTORY SET(CUSTOMER_iD) FIRST LEVEL=LEVEL_NUMBER
120 WHILE HISTORY<>”END” AND LEVEL_NUMBER<11
130 .IF PAY_DATE < “81:08:15”
140 ..DELETE HISTORY LEVEL=LEVEL_NUMBER
150 .ELSE
160 ..LEVEL_NUMBER=LEVEL_NUMBER + 1
170 .END
180 .GET HISTORY SET(CUSTOMER_iD) LEVEL=LEVEL_NUMBER
190 END
200 CONVERSE MAP
```

DELETE (RDM user view)

NOTE

This statement applies to SUPRA RDM users only.

DELETE *view-name* [ALL] [LEVEL=*level-number*]

view-name

Description	*Required*. Specifies the symbolic name of the RDM user view from which you want to delete a row.
Format	MANTIS symbolic name as defined in a previously executed VIEW statement

ALL

Description	*Optional*. Deletes all rows that would be retrieved by GET...NEXT with the same key as the row in the buffer.

LEVEL=*level-number*

Description	*Optional*. Specifies which level of MANTIS array elements contains the row you want to delete.
Default	LEVEL=1
Format	Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding VIEW statement
Consideration	You must specify LEVEL=*level-number* if the corresponding VIEW statement specifies *levels* unless you want the default value (1) or you specify ALL.

General considerations

- ♦ Before you delete a row from an RDM user view, you must first open it with a VIEW statement.
- ♦ Since RDM rows might not be uniquely keyed, you should establish your current row position in a view (by reading the row) before you execute a DELETE.
- ♦ Before deleting, you must retrieve the row by using a GET statement.
- ♦ DELETE...ALL uses the key specified by the most recent GET statement which specified the same LEVEL as the DELETE.
- ♦ You can obtain the status of the DELETE by using the “view-name“ built-in text function. The values are GOOD and HELD.

  GOOD   MANTIS successfully deleted the row.

  HELD   MANTIS failed to delete the row because it was held by another user.
- ♦ MANTIS may also return a status of NOTFOUND or ERROR, if TRAP is in effect.

  NOTFOUND   There is no row with the specified key.

  ERROR   An error occurred while deleting the row. Something may be wrong with the database, or you may have tried to perform an invalid function on the user view.
- ♦ If TRAP is not in effect and you are unable to perform the deletion because of a failure status from RDM, MANTIS automatically issues a RESET. If TRAP is in effect and the program does not issue a RESET when ERROR is returned, it is possible that MANTIS will do only part of the deletion.
- ♦ RDM Delete sets three status functions to the application program which indicate processing results: FSI, ASI, VSI. FSI indicates the success or failure of your command. ASI indicates the status of each field in the row. VSI indicates the highest field status within the row. For a complete discussion of these status functions, see “DBCS support feature” on page 455.

- ♦ Your database administrator may disallow deletions. If so, MANTIS returns the ERROR status if TRAP is in effect. Check the FSI for the reason. If TRAP is not in effect, MANTIS displays a message and halts execution.
- ♦ In an RDM user view which has levels, MANTIS has to keep track of the current row to determine which is the next row to be obtained by GET...NEXT. If a row is deleted, it can no longer be the current row. After a DELETE, the current row becomes the undeleted row with the highest-level number. If the row at every level has been deleted, the current row defaults to the row prior to the deleted row at level one.
- ♦ You can therefore read a row at each level using GET...NEXT and the next GET...NEXT retrieves the same row regardless of how many of these rows you have deleted. This is consistent with what happens in a view without levels.
- ♦ MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

Example

```
 10 VIEW CUSTOMER("CUST")
 20 SHOW "ENTER CUSTOMER NUMBER:"
 30 OBTAIN CUST_NO
 40 GET CUSTOMER(CUST_NO)
 50 IF CUSTOMER="FOUND"
 60 .DELETE CUSTOMER
 70 .SHOW"CUSTOMER DELETED"
 80 ELSE
 90 .SHOW"CUSTOMER NOT FOUND"
100 END
```

DEQUEUE

Use the DEQUEUE statement to release control of a resource (for example, a program, or a file) or, for external files release any previously reserved records. Any programs which were enqueued on the resource can then resume processing.

$$\textbf{DEQUEUE}\begin{Bmatrix} \textit{resource} \\ \textit{ultra - name} \\ \textit{access – name} \end{Bmatrix}$$

resource

Description *Optional*. Specifies the resource you want to release.

Format Text expression

ultra-name

Description *Optional*. Specifies the symbolic name of an ULTRA file view.

Format MANTIS symbolic name defined in a previously executed ULTRA statement

access-name

Description *Optional*. Specify the symbolic name of an external file.

Format MANTIS symbolic name defined in a previously executed ACCESS statement.

General considerations

- This statement is required only in very special circumstances. Consult your Master User before you use DEQUEUE.
- Use DEQUEUE *access-name* to release all locks placed on the external file by previous GET ENQUEUE statements.
- See also “ENQUEUE” on page 205 and “GET” on page 222.

♦ When *ultra-name* is specified in a DEQUEUE statement, no processing occurs because ULTRA does not provide a facility for releasing records reserved by GET...ENQUEUE before a COMMIT is performed. The specification of *ultra-name* is included for compatibility with other MANTIS systems with such a facility.

Example

```
 20 FILE REC("CUSTOMERS","ALIBABA")
 .
 .
 .
100 ENQUEUE "CUSTOMERS"+RECORD_KEY
110 GET REC(RECORD_KEY)
 .
 .
 .
200 UPDATE REC
210 DEQUEUE "CUSTOMERS"+RECORD_KEY
```

DISPLAY

Use the DISPLAY statement to display the attributes (for example, type, value) of specified MANTIS variables or MANTIS options.

$$\textbf{DISPLAY}\left\{\begin{array}{l}\textit{name}\\ \textbf{ALL}\\ \textit{options}\end{array}\right\}\textbf{,...}$$

name

Description *Required*. Specify the MANTIS variable you want to display.

Format Valid MANTIS variable name

Considerations

- If you name a complex entity (for example, a screen or file), MANTIS displays all fields defined for that entity (that is, all component variables).
- Variables in a program become defined when the program is executed.

ALL

Description ALL tells MANTIS to display all variables in your program. If you enter ALL after a variable name, MANTIS displays all variables after that variable in the program vocabulary. Variables are referenced in the vocabulary in this order:

- In the top-to-bottom occurrence of variables referenced in the program.
- In the chronological order of creation through the complex statements SCREEN, ACCESS, FILE, ULTRA, VIEW, and INTERFACE.

options

Description

Optional. Specify one of the following strings that indicates the type of MANTIS options you want to display:

String	Displays
"OPTIONS"	the normal MANTIS options, which include program parameter, memory usage, timing, screen/terminal, and SQL options.
"OPTIONS/USER"	MANTIS user options.
"OPTIONS/SECURITY"	MANTIS security options.
"OPTIONS/ALL"	all MANTIS options.

Considerations

♦ The following attributes of each MANTIS option are displayed:

- NAME. The name of the option.
- TYPE. The type of the option: Boolean, String, Numeric, or Fixed-List.
- RANGE. For String types, the minimum-maximum string length range is displayed. For Fixed-List types, the list of possible values is displayed. For Numeric types, the minimum-maximum range of values is displayed.
- VALUE. The current option value, for example, 255 (Numeric type), 5:HELLO (String type), True (Boolean type).

♦ Abbreviations are allowed. "OPT" is the shortest abbreviation for "OPTIONS," while just the first letter of "USER," "SECURITY" or "ALL" is enough to specify the type of options to be displayed. For example, the following may be used:

- DISPLAY "OPT" same as DISPLAY "OPTIONS"
- DISPLAY "OPT/U" same as DISPLAY "OPTIONS/USER"
- DISPLAY "OPT/S" same as DISPLAY "OPTIONS/SECURITY"
- DISPLAY "OPT/A" same as DISPLAY "OPTIONS/ALL"

- ♦ A comma-separated list may be specified. For example:

  ```
  DISPLAY "OPT/USER,OPT/S"
  ```

  displays the attributes of the MANTIS user options followed by those of the MANTIS security options.

General considerations

- ♦ The following attributes of each variable are displayed:

  - NAME   The name of the variable.
  - TYPE   The type of the variable: INTEGER, BIG, SMALL, DECIMAL and TEXT, and the complex variable types ENTRY, PROGRAM, SCREEN, ACCESS, FILE, ULTRA, VIEW, and INTERFACE.
  - MAXLEN   The maximum length (number of characters) of a text variable.
  - DIMS   The array dimensions if the variable is an array.
  - VALUE   If the variable is not an array, a single value is displayed, for example, 16 (numeric format), 5:HELLO (text format). For an array, values of all array elements are displayed row by row separated by commas. For file-names, interface-names, ultra-names, access-names, and view-names, the associated status is displayed. For screen-names, the value from the Reply Field is displayed.

- ♦ Your display may be as large as your full scroll map. (The size of your scroll map is determined at MANTIS start-up by the values of the SCRLROWS and SCRLCOLS MANTIS Options. You may specify the size so that it is only as big as your physical screen.) If your display is larger than your physical screen, you may need to scroll across to view the entire contents.

Examples

The following example displays the attributes of variables I, J, and K:

```
DISPLAY I,J,K
```

The following example displays the attributes of all variables in the program:

```
DISPLAY ALL
```

The following example displays the attributes of the variable J and those variables defined after J:

```
DISPLAY J,ALL
```

The following example displays the attributes of all the MANTIS options:

```
DISPLAY "OPTIONS/ALL"
```

The following example displays the attributes of the MANTIS security options:

```
DISPLAY "OPTIONS/SECURITY"
```

DO

The DO statement transfers program execution to an internal or external subroutine. An internal subroutine is a block of statements either within the existing MANTIS program, and marked by an ENTRY-EXIT. An external subroutine is identified by a PROGRAM statement. An internal subroutine performs a function required at one or more points within a program. An external subroutine performs a function required by one or more programs. After executing the subroutine and upon encountering an EXIT statement, execution returns to the next program line following the DO statement

$$\textbf{DO}\begin{Bmatrix}\textit{entry}-\textit{name}\\ \textit{program}-\textit{name}\end{Bmatrix}\quad \textbf{[(}\textit{argument}\textbf{,...)]}$$

entry-name

Description *Optional*. Specifies the name of a subroutine as defined in an ENTRY statement (for an internal subroutine).

Format MANTIS symbolic name

program-name

Description *Optional*. Specifies the name of a subroutine as defined in a PROGRAM statement (for an external subroutine).

Format MANTIS symbolic name

argument

Description *Optional*. Specifies an argument you want passed to the subroutine.

Format MANTIS symbolic name or an expression

Considerations

- ♦ You may specify up to 255 arguments.
- ♦ Each argument is either the symbolic name of a MANTIS variable or an expression:
  - If the argument is a symbolic name, each reference to the corresponding argument in the subroutine uses the specified variable (“call by reference”). Values assigned to the argument in the subroutine are stored in the specified variable.
  - If the argument is an expression, it is evaluated before calling the subroutine and each reference to the corresponding argument in the subroutine uses the value of the expression (“call by value”). Values assigned to the argument in the subroutine are lost on return from the subroutine. A subscripted symbolic name is an expression in this context. “Call by value” is not supported in MANTIS for the IBM mainframe.
- ♦ The argument(s) in the DO and ENTRY-EXIT statements must correspond in type and number.
- ♦ Since all variables previously defined by the calling routine are available to an internal subroutine, arguments are used only to set up an alias name.
- ♦ Only variables explicitly passed as arguments are available to an external subroutine. A symbolic name defined in a SCREEN, FILE, ACCESS, ULTRA, or VIEW statement may be passed (for use in CONVERSE, GET, UPDATE, INSERT, or DELETE statements), but this does not give access to the implicitly defined variables in this first group of statements.

General considerations

- You must include ENTRY-EXIT statements around a subroutine, but the entry-name on the ENTRY statement is used to DO only an internal subroutine. The *program-name* on the PROGRAM statement is used to DO an external subroutine.
- If you stop execution of a subroutine, enter SHOW DOLEVEL to see where you are. Use the UP command to return to the calling routine.
- See “Programming techniques” on page 415 for additional information on external DO.
- See also “CHAIN” on page 150, “DOLEVEL” on page 202, “ENTRY-EXIT” on page 206, and “PROGRAM” on page 308.

Examples

```
100 ENTRY EDIT_PROGRAM
 .
150 .PROGRAM EDIT_RTN("VALIDATION","COMMON")
160 .TYPE="CREDIT CHECK"
170 .DO EDIT_RTN(TYPE,CUST_NO,STATUS,MESSAGE)
180 .IF STATUS<>"GOOD"
190 ..DO ERROR_RTN(CUST_NO)
200 .END
210 .TYPE="SELECT SALES REP"
220 .DO EDIT_RTN(TYPE,CUST_NO,STATUS,MESSAGE)
230 .IF STATUS<>"GOOD"
240 ..DO ERROR_RTN(SALES_REP)
250 .ELSE
260 ..SALES_REP=MESSAGE
270 .END
280 EXIT
290 ENTRY ERROR_RTN(FIELD)
300 .IF NOTE=" "
310 ..NOTE=MESSAGE
320 ..ATTRIBUTE(MAP,FIELD)="BRI,CUR"
330 .ELSE
340 ..ATTRIBUTE(MAP,FIELD)="BRI"
350 .END
360 EXIT
```

DOLEVEL

Use the DOLEVEL function to return your current level in an external subroutine.

DOLEVEL

General consideration

- Use SHOW DOLEVEL when you are debugging to see which level you are executing. You can modify and replace programs at any level. The revised version is executed in the next RUN.

Example

```
SHOW DOLEVEL
```

E

Use the E function to return the value of the natural antilogarithm, e (2.71828183).

E

Consideration See also “EXP” on page 208 and “LOG” on page 278.

Example

```
10 X=E
20 SHOW X
```

EDIT LIST

Use the EDIT LIST statement to invoke an external full-screen editor to edit a text variable or array. The editor you get depends on the editor set by the EDITOR MANTIS Option.

EDIT LIST *text-name*

text-name

Description	*Required*. Specifies the name of the text variable or array you want to edit.
Format	Symbolic name of a text variable or array
Consideration	If you specify EDIT LIST *text-name*, the current value of the text variable or array is made available to the full-screen editor. For a scalar variable, the text occupies the only line in the editor's buffer. For an array which can have no more than one dimension, the text values of successive array elements occupy successive lines in the editor.

General considerations

- The maximum line length in the editor is 255 characters. On entry to the editor, text values that exceed the maximum line length are truncated. On exit from the editor, lines that exceed the maximum line length of the text variable or array element are truncated, excess lines are discarded, and array elements which exceed the number of output lines are set to the null string.
- You can exit from the full-screen editor with the QUIT command if you want to undo your editing.
- Use HELP OPTIONS for more information on the EDITOR MANTIS option.
- **OpenVMS** If a TPU editor is selected, the logical name TPU$SECTION is used to identify the TPU section file. If TPU$SECTION is not a defined logical name, then the logical name TPUSECINI is used instead. If an LSE editor is selected, the logical name LSE$SECTION is used to identify the LSE section file.
- **UNIX** MANTIS initiates a child process to perform the edit using a temporary file.

Example

```
TEXT SURNAMES(20,80)
EDIT LIST SURNAMES
DISPLAY SURNAMES
```

EDITRCV LIST

Use the EDITRCV LIST statement to use an external full-screen editor to recover a previously aborted EDIT statement.

NOTE

This statement applies to OpenVMS users only.

EDITRCV LIST *text-name*

text-name

Description *Required*. Specifies the name of the text variable or array you want to edit.

General considerations

- ♦ The maximum line length in the editor is 255 characters. On entry to the editor, text values that exceed the maximum line length are truncated. On exit from the editor, lines that exceed the maximum line length of the text variable or array element are truncated, excess lines are discarded, and array elements which exceed the number of output lines are set to the null string.
- ♦ You can exit from the full-screen editor with the QUIT command if you want to undo your editing.
- ♦ You may specify the external system editor that is to be used, by setting the EDITOR MANTIS Option.

Example

```
TEXT SURNAMES(20,80)
EDIT LIST SURNAMES:|
```

System crash, or you press CTRL-Y

Restart MANTIS and get back into Program Design.

```
TEXT SURNAMES(20,80)
EDITRCV LIST SURNAMES
DISPLAY SURNAMES
```

ENQUEUE

Use the ENQUEUE statement to hold control of a resource (for example, program or file). Any subsequent ENQUEUE on that resource by another program causes the program to wait until the resource is released by a DEQUEUE statement.

ENQUEUE *resource*

resource

Description *Required*. Specifies the resource you want to hold.

Format Text expression

General considerations

- ♦ MANTIS releases all enqueued resources when processing a COMMIT statement or on a terminal-read request (unless disabled by COMMIT OFF).
- ♦ This statement is required only in very special circumstances. Consult your Master User before you enqueue a resource.
- ♦ See also "GET" on page 222.

Example

```
 20 FILE REC("CUSTOMERS","ALIBABA")
  .
  .
  .
100 ENQUEUE "CUSTOMERS"+RECORD_KEY
110 GET REC(RECORD_KEY)
  .
  .
  .
200 UPDATE REC
210 DEQUEUE "CUSTOMERS"+RECORD_KEY
```

ENTRY-EXIT

The ENTRY-EXIT statement defines the boundary of a subroutine or top-level routine of a program. When a DO or CHAIN statement invokes a subroutine or program bounded by an ENTRY-EXIT, the arguments (and all references to them) passed by the DO or CHAIN statement replace the subroutine's arguments.

```
ENTRY entry-name[(argument,...)]
                .
         (statements)
                .
EXIT
```

entry-name

Description *Required*. Specifies the name of the subroutine or program.

Format MANTIS symbolic name

argument

Description *Optional*. Specifies any arguments you want passed to the subroutine or program.

Format MANTIS symbolic name

General considerations

- The *entry-name* on the ENTRY statement is used to call an internal subroutine. The *program-name* on the PROGRAM statement is used to call an external subroutine.
- Numeric and text variables and arrays should correspond in number between the ENTRY-EXIT and the DO and CHAIN statements.
- EXIT or RETURN returns control from a subroutine to the calling program, and is equivalent to the STOP statement if executed in a mainline program.

- See “Programming techniques” on page 415 for more details on this statement.
- See also “DO” on page 199, “CHAIN” on page 150, “PROGRAM” on page 308, “RETURN” on page 320, “STOP” on page 349.
- Internal subroutines have access to all the variables defined in the surrounding program context. Be careful not to confuse ENTRY statement arguments with global program variables, since the global variables having the same name as ENTRY arguments are not accessible within the scope of the subroutine.
- Must be a unique symbolic name within the program. The entry name is defined at the time the program is saved or replaced, that is, before any statements are executed. For example,

```
00010 BIG X

…

01000 ENTRY X

…

02000 EXIT
```

- X will be defined as the entry name. Statement 10 will be ignored because X is an already defined symbolic name.

Example

```
 10 ENTRY DATA_ENTRY
 20 .SCREEN MAP1("INDEX")
 30 .FILE REC1("INDEX","SERENDIPITY")
 40 .CONVERSE MAP1
 50 .WHILE MAP1<>"CANCEL"
 60 ..DO INSERT_RECORD
 70 ..CLEAR MAP1
 80 ..CONVERSE MAP1
 90 .END
100 .STOP
110 EXIT
120 ENTRY INSERT_RECORD
130 .INSERT REC1
140 EXIT
```

EXP

Use the EXP function to return the value of natural e to the power of an arithmetic expression.

EXP(*a*)

a

Description *Required*. Specifies a power for the function.

Format Arithmetic expression

Considerations

- ♦ The expression result is coerced to BIG. Decimal expression results may therefore lose some precision or may generate an arithmetic fault (310) if the magnitude is too large for a BIG data type.
- ♦ See also “E” on page 202 and “LOG” on page 278.

Example

```
 120 Z=100
 130 X=EXP(Z)
 140 SHOW X

 RUN
results in:
 2.6881171E43
```

FALSE

Use the FALSE function to return the value FALSE. FALSE returns a value of 0.

FALSE

General considerations

- Any numeric expression that evaluates to zero can be considered FALSE for purposes of a logic statement (IF, UNTIL, WHEN, and WHILE). For example, the expression NUMBER_OF_ERRORS will evaluate to FALSE when NUMBER_OF_ERRORS is zero, and will be true otherwise.
- See also “TRUE” on page 362 and “ZERO” on page 402.

Example

```
 10 ENTRY CUST_ENTRY
 20 .SCREEN MAP("CUST_ENTRY")
 30 .FILE REC("CUST_FILE",PASSWORD)
 40 .FILE ST_CODES("STATE_CODES",PASSWORD)
 50 .CONVERSE MAP
 60 .WHILE MAP<>"CANCEL"
 70 ..ERROR=FALSE
 80 ..DO VALIDATE_INFO
 90 ..IF NOT(ERROR)
100 ...INSERT REC
110 ...CLEAR MAP
120 ..END
  .
  .
  .
```

FILE

Use the FILE statement to define a MANTIS file that your program accesses. MANTIS retrieves the file profile from the library and defines a MANTIS variable or array with the same name as each field in the file profile.

FILE *file-name*(*libname*,*password*[,PREFIX][,*levels*]),...

file-name

Description	*Required*. Specifies the name for the file that you use in subsequent GET, UPDATE, INSERT, and DELETE statements.
Format	MANTIS symbolic name
Consideration	No processing occurs if *file-name* is already defined.

libname

Description *Required*. Specifies the name of the file profile as it was saved during MANTIS File Design.

Format Text expression that evaluates to a library name in the format:

```
[user-name:] file-name
```

Consideration If the file profile is in your own library, you do not need to specify *user-name*.

password

Description	*Required*. Specifies the password needed for the type of file access (GET, UPDATE, INSERT/DELETE) your program requires.
Format	Text expression that evaluates to the password specified during MANTIS File Design
Consideration	The file DATA can be in the Shared Entity Pool and/or the MANTIS File. In order to access the shared DATA set, you must specify the read-only access password. Unless the FILE statement enforces read-only access, the DATA on the MANTIS File will be accessed in preference to the DATA set loaded in the Shared Pool. It is not possible to access some records from the Shared Pool and others from the MANTIS File—you get one or the other.

PREFIX

Description *Optional*. Indicates that MANTIS place the prefix "*file-name*_" on all variable names associated with this file. For example, if the MANTIS file BOTTLES has a field named VOLUME, the following statement causes the corresponding variable BIN_VOLUME to be defined in the work area:

```
10 FILE BIN("BOTTLES","MAGIC",PREFIX)
```

levels

Description *Optional*. Specifies the number of levels you want to use in GET, UPDATE, INSERT, and DELETE statements for this file.

Format Arithmetic expression that evaluates to a value in the range 1–255

Considerations

- If you specify *levels*, you must also specify LEVEL=*level-number* (unless you want the default LEVEL=1) in all GET, UPDATE, INSERT, and DELETE statements for the file.
- If you do not specify *levels*, MANTIS defines an INTEGER, BIG, SMALL, DECIMAL or TEXT variable or array for each field in the file profile according to the type and dimensions specified during MANTIS File Design.
- If you specify *levels*, MANTIS adds an extra dimension to each field to allow a separate value to be stored at each level. MANTIS defines a one-dimensional array instead of a variable, a two-dimensional array instead of a one-dimensional array, and so on. The first dimension of each array is *levels*.

  OpenVMS If the MANTIS File is RU Journal , all internal MANTIS files are automatically RU Journal .

Examples

```
 20 .FILE RECORD("INDEX","SERENDIPITY",16)
 30 .SCREEN MAP("INDEX")
 40 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
 50 ..CLEAR MAP:LEVEL_NUMBER=1
 70 ..GET RECORD LEVEL=LEVEL_NUMBER
 80 ..WHILE RECORD<>"END" AND LEVEL_NUMBER < 16
 90 ...LEVEL_NUMBER=LEVEL_NUMBER+1
100 ...GET RECORD LEVEL=LEVEL_NUMBER
110 ..END
120 ..CONVERSE MAP
130 .END
```

```
 10 FILE RECORD("EXAMPLE:INDEX","SERENDIPITY",PREFIX,16)
 20 TEXT PASS(16)
 30 TEXT FILENAME(33)
 40 PASS="SERENDIPITY"
 50 FILENAME="EXAMPLE:INDEX"
 60 FILE RECORD(FILENAME,PASS)
```

file-name

Use the file-name function to return a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on a MANTIS file with the symbolic name "*file-name*".

file-name

file-name

Description *Required*. Specifies the symbolic name of the MANTIS file.

Format MANTIS symbolic name as defined in a previously executed FILE statement

Example

```
200 ENTRY VALIDATE_INFO
210 .MESSAGE=""
220 .FILE CUSTOMER_FILE("CUST_FILE",PASSWORD,PREFIX)
230 .GET CUSTOMER_FILE(CUST_NUMBER)
240 .IF CUSTOMER_FILE="FOUND"
250 ..ATTRIBUTE(MAP,CUST_NUMBER)="BRI,CUR"
260 ..MESSAGE="DUPLICATE CUSTOMER NUMBER"
270 .END
 .
 .
 .
```

FOR-END

Use the FOR-END statements to execute a block of statements repeatedly while a counter is incremented or decremented through a specified range of values.

FOR *counter* **=** *initial* **TO** *final* **[BY** *increment* **]**

.

statements

.

END

counter

Description	*Required*. Specifies the numeric variable or array element that MANTIS uses as a counter.
Format	A MANTIS symbolic name (see “MANTIS symbolic names” on page 65).

initial

Description	*Required*. Specifies the initial value of the counter.
Format	Arithmetic expression

final

Description	*Required*. Specifies the final value with which the counter is compared.
Format	Arithmetic expression

increment

Description	*Optional*. Specifies the increment to be added to the counter after each execution of the block of statements..
Default	1
Format	Arithmetic expression

General considerations

- ♦ MANTIS sets the counter to the initial value which is evaluated only once. MANTIS evaluates the final value and the increment prior to each execution of the block of statements.
- ♦ MANTIS repeatedly executes the block of statements while the counter is less than or equal to the final value and the increment is positive, or the counter is greater than or equal to the final value and the increment is negative. MANTIS will also stop executing the block of statements when the increment evaluates to 0. After executing each block of statements, MANTIS adds the increment to the counter.
- ♦ MANTIS will not execute the block of statements when the initial value exceeds the final value (when the increment is positive), or the initial value is less than the final value (when the increment is negative).
- ♦ If the block of statements changes the values of the counter, the final value, or the increment, it is reflected in the next comparison of the counter and the final value.
- ♦ For efficiency, if the final value or increment are expressions that do not change within the loop, assign them to a symbolic name so that they do not need to be evaluated on each iteration of the loop. (See example below using FINAL)
- ♦ See also “BREAK” on page 146, “NEXT” on page 286, “WHEN-END” on page 399, “WHILE-END” on page 401, “UNTIL-END” on page 373.

- The logic of FOR-END statements can be expressed in terms of equivalent WHILE-END statements as follows:

```
10 FOR COUNTER=INITIAL TO FINAL BY INCREMENT
20 .SHOW COUNTER
30 END
40 COUNTER=INITIAL
50 WHILE INCREMENT>0 AND COUNTER<=FINAL
60 .'OR INCREMENT<0 AND COUNTER>=FINAL
70 .SHOW COUNTER
80 .COUNTER=COUNTER + INCREMENT
90 .END
```

Examples

This shows the value of each element of an array.

```
10 FINAL=SIZE(ARRAY,1)
20 FOR I=1 TO FINAL
30 .SHOW ARRAY(I)
40 END
```

This example checks a string for any occurrence of “/”, “\” or “]” and returns J as the right most position where any was found or zero (0) if none were found.

```
10 FOR J=SIZE(STRING) TO 1 BY -1
20 .IF STRING(J,J)="/" OR STRING(J,J)="\" OR STRING(J,J)="]"
30 ..BREAK
40 .END
90 END
```

FORMAT

Use the FORMAT function to convert a specified number into a particular display using an edit mask. FORMAT is useful for creating reports using the SHOW statement. The masked conversion done by FORMAT is identical to that done by the CONVERSE statement where each field's edit mask is stored with the screen design. The exception is, only the MANTIS defaults of DECIMAL "." and/or THOUSANDS "," should be used within the mask for these edit characters. MANTIS will convert these characters to the current MANTIS Options setting for the THOUSANDS and DECIMAL Options if needed.

FORMAT(*number*, *mask* [, *mask-character*])

number

Description *Required*. Specifies the numeric expression you want to format with an edit mask.

Format Arithmetic expression

mask

Description *Required*. Specifies the edit mask.

Format Text expression or text literal

Consideration For more information about edit masks, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.

mask-character

Description *Optional*. Specify the character in the mask operand that is to be filled with digits from the supplied number.

Default The program's mask character, which is inherited from the FMTMASK MANTIS Option when the program is created (and may be altered in the Program Design Facility, Library Functions). The default value for the FMTMASK MANTIS Option is the hash character (#).

Format Text expression whose first character specifies the mask character.

Consideration If the specified mask character is the NULL string, the program's mask character is used in the interpretation of the mask parameter (that is, as if no mask-character parameter had been specified).

General considerations

- For more information on numeric edit masks, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.
- See also "TXT" on page 363.

Examples To test your screen design masks for expected results, the following code:

```
PHONE=5136622300
PHONE_NO=FORMAT(PHONE,"###-###-####")
```

produces this result:

```
513-662-2300
```

To provide formatted output for SHOW fields or fields on a screen or file which are not normally formatted, the following code:

```
10  SHOW NAME; AT(40)FORMAT(ACCT_NO,"Z##-####-#")AT(55)
20  'FORMAT(BALANCE,"##;##Z.##CR")
```

produces this result when the CR and DECIMAL MANTIS Options take on their default values:

```
HENRIETTA JOHNSON  034-4783-1  127.89CR
```

The use of the FORMAT function mask character is shown in the following code:

```
10 PART_NO=38765
20 SHOW FORMAT(PART_NO, "PART # ?????", "?")
```

which produces this result:

```
PART # 38765
```

The use of the FORMAT function mask characters is shown in the following code example, with the DECIMAL and THOUSANDS Options set as not using the MANTIS default values:

```
10 SET $OPTION(“DECIMAL=:”)
20 SET $OPTION (“THOUSANDS=.”)
30 SET $OPTION(“DECIMAL=,”)
40 TEXT TXT_DATA
50 BIG NUM_DATA
60 TXT_DATA=”3456,78”
70 NUM_DATA=VALUE(TXT_DATA)
80 SHOW FORMAT (NUM_DATA, “NUM_DATA= #,###.##”)
```

Which produces this result:

```
NUM_DATA= 3.456,78
```

FSI

Use the FSI function to indicate the success or failure of an RDM user view, MANTIS file, or external file GET, DELETE, INSERT or UPDATE.

FSI(*name*[,*message*])

name

Description	*Required*. Specifies the symbolic name by which you refer to the user file or view.
Format	MANTIS symbolic name as defined in a previously executed VIEW, FILE, or ACCESS statement.

message

Description	*Optional*. Specifies a text variable to receive an error message.
Format	MANTIS symbolic name
Consideration	The symbolic name must identify a text variable with a length of at least 40.

General considerations

- Function Status Indicators (FSIs) reflect the success or failure of the most recent RDM operation on the specified user view. See “DBCS support feature” on page 455 for more details on FSIs and user views.
- For MANTIS file and external file DELETEs that succeed in deleting one or more records, the number of records deleted is returned in the message area of the FSI in the format “nnnnn DELETED RECORDS.” All asterisks in the nnnnn portion indicate a record number greater than five characters.
- See also “ASI (statement)” on page 107, “VSI( )” on page 396 and “view-name” on page 395.

Example

The following example shows how to use the FSI function to test for a successful get.

```
 20 VIEW CUSTOMERS("NEW_CUSTOMERS")
  .
 50 TEXT FSI_MESSAGE(40)
  .
  .
110 GET CUSTOMERS
120 IF FSI(CUSTOMERS)<>"GOOD"
  .
  .
150 SHOW "FSI status  = ";FSI(CUSTOMERS,FSI_MESSAGE)
160 SHOW "FSI message = ";FSI_MESSAGE
```

The following example shows how to use the FSI function to test for a specific value and have MANTIS return an error message:

```
00010 ACCESS X ("TEXT","PSW")
00020 TEXT MESSAGE(40)
00030 TRAP X ON
00040 GET X FIRST
00050 IF X="ERROR"
00060 .IF FSI(X,MESSAGE)="UNAVAILABLE"
00070 ..SHOW "FILE TEST IS UNAVAILABLE"
00080 .ELSE
00090 ..SHOW "FILE TEST GET FAILED:STATUS="+MESSAGE
00100 .END
00105 .WAIT
00120 END
```

The following example shows the debugging use of FSI:

```
COMMAND ===> SHOW FSI(X,MESSAGE), MESSAGE
```

GET

Use the GET statement to read a record from a MANTIS file, an external file, an ULTRA file, or a row from an RDM user view. MANTIS copies the fields in the record or the row to the corresponding MANTIS variables and arrays. Before you can read a record from a file or a row from a user view, you must open it by processing the associated FILE, ACCESS, ULTRA, or VIEW statement.

GET (MANTIS file)

$$\textbf{GET}\ \textit{file} - \textit{name}\ \left[\begin{array}{l} \textbf{(}\textit{key},\ldots)\left[\begin{array}{l}\textbf{NEXT}\\ \textbf{EQUAL}\end{array}\right]\\ \textbf{FIRST}\\ \textbf{LAST}\\ \textbf{PRIOR}\\ \underline{\textbf{NEXT}} \\ \end{array}\right] \textbf{[ENQUEUE][LEVEL} = \textit{level} - \textit{number}\ \textbf{]}$$

file-name

Description *Required*. Specifies the symbolic name of the MANTIS file you want to access.

Format MANTIS symbolic name as defined in a previously executed FILE statement

$$\textbf{(}\textbf{\textit{key}}\,\textbf{,...)}\left[\begin{array}{l}\textbf{NEXT}\\ \textbf{EQUAL}\end{array}\right]$$

Description *Optional*. Specifies the key of the desired record. MANTIS concatenates the supplied key values into a single access key. Each key parameter must match the data type of the corresponding KEY field in the FILE design. If you omit trailing key fields the resulting partial key is used to access the file.

Format Expression of the same type and size as the corresponding key element in the MANTIS file profile

Considerations

- ♦ The default key-matching criterion is "greater or equal." Use NEXT or EQUAL to set alternate key-matching criteria. NEXT specifies "greater"; EQUAL specifies "equal."
- ♦ If you specify EQUAL, all key values in the retrieved record must be identical to corresponding supplied key values, or a status of "NOTFOUND" is returned. You should supply a full key, not a partial key, with the EQUAL parameter.

FIRST

Description *Optional*. MANTIS repositions to the beginning of the MANTIS file and retrieves the first record in the file. The return status is "NEXT" if a record is returned, or "END" if the file is empty.

LAST

Description *Optional*. MANTIS repositions to the end of the MANTIS file and retrieves the last record in the file. The return status is "NEXT" if a record is returned, or "END" if the file is empty.

Consideration LAST functionality is not implemented. It is only included to provide syntax compatibility with MANTIS for the IBM mainframe.

PRIOR

Description *Optional*. Retrieves the previous record in the MANTIS file and gives a return status of "NEXT." If no position exists in the file, the LAST record is returned with a return status of "NEXT." If the beginning of the file is reached the return status is "END."

Consideration PRIOR functionality is not implemented. It is only included to provide syntax compatibility with MANTIS for the IBM mainframe.

NEXT

Description *Optional*. Retrieves the next record in the MANTIS file. NEXT is the default option for the GET statement.

ENQUEUE

Description *Optional*. Tells MANTIS to obtain a lock on the record.

Consideration This prevents the record from being accessed by another user (but not by another MANTIS File View Description of the same MANTIS file).

LEVEL=*level-number*

Description *Optional*. Specifies which level of MANTIS array elements the fields in the record are copied to.

Default LEVEL=1

Format Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding FILE statement

Consideration You must specify LEVEL=*level-number* if the corresponding FILE statement specifies *levels* unless you want the default value.

General considerations

- Before you can read a record from a MANTIS file, you must open it with a FILE statement.
- MANTIS reads the record and copies each field into the corresponding MANTIS variable or array element. You can obtain the status of the GET by using the “file-name“ built-in text function. The status can be one of the following values:

  FOUND MANTIS successfully retrieved the record identified by the key you supplied.

  NEXT MANTIS retrieved the next record in sequence for a GET without a key, or MANTIS retrieved the next record in key sequence when it could not locate the supplied key or when a partial key was supplied.

  END MANTIS failed to retrieve the record because it reached the end of the file.

  NOTFOUND EQUAL was specified and MANTIS could not find a record with an equal key.

  HELD MANTIS failed to retrieve the record because it was held by another user.

- MANTIS may also return a status of LOCK, ERROR, or DATA if TRAP is in effect.

  LOCK   The password specified in the FILE statement is invalid or the record is locked by another user.

  ERROR   A physical error occurred while retrieving the record.

  DATA   The record retrieved contained an invalid number.

- If the GET is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the system error message that may have been returned. This information is also available to MANTIS fault handling routines (see the SET TRAP statement).

- You may specify fewer key parameters than there are key elements in the file profile. MANTIS then retrieves the first record in which the corresponding key elements are equal to or greater than the partial key. When you use a partial key, MANTIS returns a status of NEXT unless it reaches the end of file (when it returns a status of END).

- GET...(key,...) retrieves the record with the specified key. GET...FIRST retrieves the first record (in key sequence) in the file. GET...[NEXT] retrieves the next record (in key sequence) in the file.

- If you issue GET...EQUAL, a subsequent GET...NEXT returns the next record (in key sequence), even if a NOTFOUND status was returned on the GET...EQUAL.

- Records locked using ENQUEUE are released by a COMMIT statement, by automatic COMMIT processing (if enabled) on the next terminal read, or by MANTIS closing the MANTIS File on program termination.

Example

```
 20 .FILE RECORD("INDEX","SERENDIPITY",16)
 30 .SCREEN MAP("INDEX")
 40 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
 50 ..CLEAR MAP:LEVEL_NUMBER=1-
 70 ..GET RECORD("WILLIAMS") LEVEL=LEVEL_NUMBER
 80 ..WHILE RECORD<>"END" AND LEVEL_NUMBER < 16
 90 ...LEVEL_NUMBER=LEVEL_NUMBER+1
100 ...GET RECORD LEVEL=LEVEL_NUMBER
110 ..END
120 ..CONVERSE MAP
130 .END
```

GET (external file)

$$\textbf{GET}\ \textit{access} - \textit{name}\ \begin{bmatrix} \textbf{(}\textit{key},\dots\textbf{)}\begin{bmatrix}\textbf{NEXT}\\ \textbf{EQUAL}\end{bmatrix} \\ \textbf{FIRST} \\ \textbf{LAST} \\ \textbf{PRIOR} \\ \underline{\textbf{NEXT}} \\ \end{bmatrix} \textbf{[ENQUEUE][LEVEL} = \textit{level} - \textit{number}\,\textbf{]}$$

access-name

Description *Required*. Specifies the symbolic name of the external file you want to access.

Format MANTIS symbolic name as defined in a previously executed ACCESS statement

Consideration If the file has been released, it is automatically reopened, but the position in the file is lost. For example, if a GET...NEXT is subsequently executed, the first record in the file is returned regardless of which record was being accessed before the release.

$$\textbf{(}\textbf{\textit{key}}\,\textbf{,...)}\begin{bmatrix}\textbf{NEXT}\\ \textbf{EQUAL}\end{bmatrix}$$

Description *Optional*. Specifies the key of the desired record. MANTIS concatenates the supplied key values into a single access key. Each key parameter must match the data type of the corresponding KEY field in the external file view. If you omit trailing key fields the resulting partial key is used to access the file.

Format Expression of the same type and size as the corresponding key element in the external file view.

Considerations

- The default key-matching criterion is “greater or equal.” Use NEXT or EQUAL to set alternate key-matching criteria. NEXT specifies “greater”; EQUAL specifies “equal.”
- If you specify EQUAL, all key values in the retrieved record must be identical to corresponding supplied key values, or a status of “NOTFOUND” is returned. You should supply a full key, not a partial key, with the EQUAL parameter.

FIRST

Description *Optional*. MANTIS repositions to the beginning of the external file and retrieves the first record in the file. The return status is “NEXT” if a record is returned, or “END” if the file is empty.

LAST

Description *Optional*. MANTIS repositions to the end of the external file and retrieves the last record in the file. The return status is “NEXT” if a record is returned, or “END” if the file is empty.

Consideration LAST functionality is not implemented. It is only included to provide syntax compatibility with MANTIS for the IBM mainframe.

PRIOR

Description *Optional*. Retrieves the previous record in the external file and gives a return status of “NEXT.” If no position exists in the file, the LAST record is returned with a return status of “NEXT.” If the beginning of the file is reached the return status is “END.”

Consideration PRIOR functionality is not implemented. It is only included to provide syntax compatibility with MANTIS for the IBM mainframe.

NEXT

Description *Optional*. Retrieves the next record in the external file. NEXT is the default option for the GET statement.

ENQUEUE

Description *Optional*. Tells MANTIS to obtain a lock on the record.

Consideration This prevents the record from being accessed by another user or by another view of the same external file. In other words, GET and GET ENQUEUE will return HELD.

LEVEL=*level-number*

Description *Optional*. Specifies the level of MANTIS array elements where the fields in the record are copied.

Default LEVEL=1

Format Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ACCESS statement

Consideration You must specify LEVEL=*level-number* if the corresponding ACCESS statement specifies *levels* unless you want the default value.

General considerations

- Before you can read a record from an external file, you must open it with an ACCESS statement.
- For indexed files, the key you specify in the GET statement must correspond to the key element(s) you specified during External File View Design. If an indexed file has alternate keys defined, you specify the key of reference during External File View Design.
- For sequential files, the key is the Relative File Address (RFA) -- TEXT(6). MANTIS returns the RFA of the retrieved record in the associated reference variable if one was defined during External File View Design.
- For relative files, the key is the Relative Record Number (RRN) -- BIG precision. (Refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.) The first record has an RRN of 1. MANTIS returns the RRN of the retrieved record in the associated reference variable defined during External File View Design.

- MANTIS reads the record and copies each field into the corresponding MANTIS variable or array element. You can obtain the status of the GET by using the “access-name“ built-in text function. The status can be one of the following values:

  FOUND MANTIS successfully retrieved the record identified by the key you supplied.

  NEXT MANTIS retrieved the next record in sequence for a GET without a key, or MANTIS retrieved the next record in key sequence when it could not locate the supplied key or when a partial key was supplied.

  END MANTIS failed to retrieve the record because it reached the end of the file.

  NOTFOUND EQUAL was specified and MANTIS could not find a record with an equal key.

  HELD MANTIS failed to retrieve the record because it was held by another user.

- MANTIS may also return a status of LOCK, ERROR, or DATA if TRAP is in effect.

  LOCK The password specified in the ACCESS statement is invalid or the record is locked by another user.

  ERROR A physical error occurred while retrieving the record. The RFA for the sequential file was not at a record boundary or the RRN for a relative file specified a record number outside the file range.

  DATA The record retrieved contained an invalid number.

- If the GET is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the system error message that may have been returned. This information is also available to MANTIS fault handling routines (see the SET TRAP statement).

- If you issue GET...EQUAL, a subsequent GET...NEXT returns the next record (in key sequence) even if a NOTFOUND status was returned on the GET...EQUAL.

- ♦ Records locked using ENQUEUE are released by a COMMIT statement, by automatic COMMIT processing (if enabled) on the next terminal read or when the external file is closed by a RELEASE statement (if it is not delayed) or by program termination.
- ♦ Note that multiple keys may only be specified for indexed files with segmented keys.

Example

```
20 ACCESS RECORD("INDEX","SERENDIPITY",16)
30 SCREEN MAP("INDEX")
40 CONVERSE MAP
50 COUNTER=1
60 WHILE MAP<>"CANCEL" AND COUNTER<17
70 .GET RECORD LEVEL=COUNTER
80 END
```

GET (ULTRA file view)

NOTE: This statement applies to SUPRA PDM users only.

$$\textbf{GET}\ \textit{ultra}-\textit{name}\ \begin{bmatrix}(\textit{key},\ldots)\\ \\ \textbf{SET}\ (\textit{key},\ldots)\\ \\ \textbf{FIRST}\end{bmatrix}\begin{bmatrix}\textbf{AT} & \textit{refer}\\ \textbf{BEFORE} & \textit{refer}\\ \underline{\textbf{AFTER}} & \textit{refer}\\ \textbf{FIRST} & \\ \textbf{LAST} & \end{bmatrix}$$

$$[\textbf{ENQUEUE}][\textbf{LEVEL} = \textit{level}-\textit{number}\,]$$

ultra-name

Description	*Required*. Specifies the symbolic name of the ULTRA file you want to access.
Format	MANTIS symbolic name as defined in a previously executed ULTRA statement

(*key*,…)

Description	*Optional*. Specifies the key of the desired record in a primary file. MANTIS concatenates the supplied key values into a single access key. Each key parameter must match the data type of the corresponding KEY field in the ULTRA file view. All key fields must be supplied—partial keys are not allowed and cause a return status of “NOTFOUND.”
Format	Expression of the same type and size of the corresponding key element in the ULTRA file profile.

SET(*key*,…)

Description	*Optional*. Specifies a set or a chain of records in a related file. The records belonging to the set do not have individual keys by which you can refer to them. Instead, a key identifies the whole set. You can retrieve individual records in the set by specifying their position relative to a reference variable.
Consideration	This parameter applies to related files only.

AT
BEFORE
AFTER
FIRST
LAST

Description	*Optional*. Specifies the relative position of a record in a set.
Default	AFTER the record specified by the reference variable
Options	AT selects the record specified by refer.
	BEFORE selects the record before the one specified by refer
	AFTER selects the record after the one specified by refer
	FIRST selects the first record in the set
	LAST selects the last record in the set

refer

Description	*Optional*. Identifies the position in a related file AT, BEFORE, or AFTER which you want to retrieve a record.
Format	Four-character text expression
Consideration	The refer value is usually the value of the reference variable.

FIRST

Description	*Optional*. MANTIS repositions to the beginning of the ULTRA file and retrieves the first record in the file. The return status is “NEXT” if a record is returned, or “END” if the file is empty.

ENQUEUE

Description *Optional*. Tells MANTIS to lock out all other users' modifications to the ULTRA record until the record being retrieved has been updated, deleted, or dequeued by a COMMIT statement.

LEVEL=*level-number*

Description *Optional*. Specifies the level of MANTIS array elements where the fields in the record are copied.

Default LEVEL=1

Format Arithmetic expression which evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ULTRA statement

Consideration You must specify LEVEL=*level-number* if the corresponding ULTRA statement specifies *levels* unless you want the default value.

General considerations

- Before you can read a record from an ULTRA file, you must open it with an ULTRA statement.
- MANTIS retrieves the record from the file and assigns the data to MANTIS variables according to the conversion rules defined in your ULTRA file view. For more details, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.
- When you use the SET parameter, MANTIS stores the current position in the reference variable specified during ULTRA File View Design. For more details, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300. You may save and restore this position using a text variable of at least four characters in length.

- You can obtain the status of the operation by using the “ultra-name” built-in text function. The values are END, NEXT, FOUND, and NOTFOUND.

  END MANTIS failed to retrieve the record because it reached the end of the file or set.

  NEXT MANTIS retrieved the next record in sequence because you issued the GET statement without a key.

  FOUND MANTIS successfully retrieved the record identified by the supplied key.

  NOTFOUND The primary file record identified by the supplied key does not exist.

- MANTIS may also return a status of LOCK, ERROR, DATA, or HELD if TRAP is in effect.

  LOCK The password specified in the ULTRA statement for this file view is not valid.

  ERROR An error occurred while retrieving the record.

  DATA The record retrieved contained an invalid number.

  HELD MANTIS failed to retrieve the record because it was held by another user. This situation normally calls for a finite number of retries of the GET statement, since individual records would not normally be held for long. Note, in IBM compatibility mode, HELD is not returned, and ERROR is returned for this condition

- If MANTIS returns a status of END after serially reading through a primary or related file chain, the next GET retrieves the first record of the file or chain.

Example

```
 50 ULTRA CUSTOMERS(“CLIENT”,”SALES”)
 60 ULTRA HISTORY(“PAYMENTS”,”TEXAS”,11)
 70 TEXT CUSTOMER_iD(20)
 80 SMALL LEVEL_NUMBER
 90 CUSTOMER_iD=”OUR-BEST”
100 GET CUSTOMERS(CUSTOMER_iD)
110 LEVEL_NUMBER=1
120 GET HISTORY SET(CUSTOMER_iD)FIRST LEVEL=LEVEL_NUMBER
130 WHILE HISTORY<>”END” AND LEVEL_NUMBER<11
140 .LEVEL_NUMBER=LEVEL_NUMBER+1
150 .GET HISTORY SET(CUSTOMER_iD)AFTER
155 .’REFER(LEVEL_NUMBER-1)LEVEL=LEVEL_NUMBER
160 END
170 CONVERSE MAP
```

GET (PDM index get)

$$\textbf{GET}\ \textit{ultra-name}\ \left[\textbf{(}\textit{key1,key2...keyn}\textbf{)}\right]\left\{\begin{array}{l}\textbf{KEYF}\\ \textbf{KEYR}\\ \textbf{KEYM}\\ \textbf{KEYS}\end{array}\right\}\textit{index–id}$$

$$\left[\textbf{MASK=}\textit{m}\right]\left[\textbf{ENQUEUE}\right]\left[\textbf{LEVEL=}\textit{level-number}\right]$$

ultra-name

Description	*Required*. Specifies the name of the PDM file you want to access.
Format	MANTIS symbolic name as defined in a previously executed ULTRA statement

(key1, key2…keyn)

Description *Conditional*. Specifies the record key(s) of the desired record. *Optional* for KEYR and KEYF. *Required* for the *first* KEYS and KEYM.

Format MANTIS symbolic names as defined in the ULTRA file view record layout.

Considerations

- The variable(s) that make up the key value must be elements of the file view specified in the GET statement.

- For KEYF and KEYR, PDM uses the key value to initialize its search. If the key value is not given for KEYF, MANTIS supplies a zero-length key, causing the PDM to begin its search at the beginning of the key. Subsequent GETs with KEYF will read down the key. If the key value is not given for KEYR, MANTISsupplies a zero-length key, causing the PDM to start at the end of the key. Subsequent GETs with KEYR will read up the key.

- For KEYS and KEYM, *keyn* is required for the first GET because it must contain the mask value for which the PDM is looking. The key value causes PDM to scan the key in ascending sequence and returns those key values which meet the criteria of the supplied "masked" key value. The key value must be omitted on subsequent GET with KEYM/KEYS in order to continue the sequential scan of the specified index.

$$\left\{\begin{matrix} \mathit{KEYF} \\ \mathit{KEYR} \\ \mathit{KEYM} \\ \mathit{KEYS} \end{matrix}\right\}$$

Description *Required*. Specifies the type of GET performed.

Options KEYF *Key Forward* processing retrieves records in ascending sequential order

KEYR *Key Reverse* processing retrieves records in descending sequential order

KEYM *Key Mask* processing scans the index in ascending sequence and returns only those key values which meet the criteria of the masked key value supplied. The bound qualifier will be used in subsequent GETs using KEYM.

NOTE

In KEYM mode, the specified key-mask is matched against actual keys retrieved from the index and the record is accepted if the two keys match from the beginning up to the length of the supplied key-mask. The key-mask must contain at least one wildcard character as specified by the mask character *m*. See the General considerations and Examples for more information on KEYM.

KEYS *Key Search* scans the index in ascending sequence and returns only those key values which contain any occurrence of the search key value. The bound qualifier will be used in subsequent GETs using KEYS.

NOTE

In KEYS mode, the pattern-match is not anchored to the beginning of the actual key, which is scanned from left to right until the first occurrence of the search-key is successfully matched. The search key need not contain a wildcard character.See the General considerations and Examples for more information on KEYS.

index-id

Description	*Required*. Identifies the PDM secondary key used to access the data set.
Format	text expression of at least two characters.
Consideration	The last two characters of the index-id are taken as the secondary key name used to access the data set.

MASK=*m*

Description	*Required* when using KEYM and KEYS when a key value is supplied.
Format	Text expression of at least one character.

Considerations

- The first character of the mask expression result specifies the wildcard character in a key-mask associated with KEYM or in a search key associated with KEYS.
- A wildcard character in the supplied key-mask or search key matches any character in the corresponding position in an actual key retrieved from the index.

ENQUEUE

Description	*Optional*. Tells MANTIS to lock out all other users' modifications to the ULTRA record until the record being retrieved has been updated, deleted, or dequeued by a COMMIT statement.

LEVEL=*level-number*

Description	*Optional*. Specifies the buffer number that will receive the record you want to retrieve.
Default	LEVEL=1
Format	Arithmetic expression which evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ULTRA statement.
Consideration	You must specify LEVEL=*level-number* if the corresponding ULTRA statement specifies *levels*, unless you want the default value.

General considerations

- You can obtain the status of the operation by using the ultra-name built-in text function. The possible values are shown below.

  KEYF/KEYR:

  FOUND MANTIS found an actual key which exactly matches the supplied key value.

  NEXT MANTIS did not find an exact match of the key value but retrieved the record that was next in sequence.

  NOINDEX The PDM rejected the specified *index-id*. No such secondary key exists in the DBMOD for the data set.

  END MANTIS reached the end of the index.

  KEYM/KEYS:

  NEXT MANTIS retrieved the next record matching the key-mask or search key criteria.

  NOMASK The supplied key-mask does not contain a wildcard character as specified by the MASK expression *m*.

  NOINDEX The PDM rejected the specified *index-id.* No such secondary key exists in the DBMOD for the data set.

  END MANTIS reached the end of the index.

- MANTIS may also return a status of LOCK, ERROR, DATA or HELD if TRAP is in effect.

  LOCK The password specified in the ULTRA statement for this file view is not valid.

  ERROR An error occurred while retrieving the record.

  DATA The record retrieved contained an invalid number.

  HELD MANTIS failed to retrieve the record because it was held by another user. This situation normally calls for a finite number of retries of the GET statement, since individual records would not normally be held for long. Note, in IBM compatibility mode, HELD is not returned, and ERROR is returned for this condition.

- *Bound qualifier* is a term used in explaining how PDM index retrieval works. The PDM returns a bound qualifier on the first successful index GET. While the qualifier is bound it contains a valid "current position" within the index. The qualifier will remainbound on subsequent GETs until one of the following happens:

  - A status of END is returned.
  - Another type of index GET is executed, for example, KEYR after KEYF.
  - The index-id or ultra-name is changed.
  - A key value is specified in the index GET statement.
  - An EXIT, STOP or CHAIN statement is executed.

- You may want to save your key value before the GET. It will be overlaid by the returned key value since the elements are defined in the ULTRA file view.

- For KEYM and KEYS, wildcard pattern matching is restricted where numeric key fields are specified in the key. Since you cannot place text wildcard characters in a numeric variable, MANTIS allows you to flag the variable as a generic key field by following it with ',=' in the key list. This means that all numbers in that field of the actual key will match the supplied key-mask. For example, if BMPQTY is numeric and BMPART is text, use ",=," to get the mask character extended to BMPQTY, as follows:

  ```
  GET MAS (BMPQTY,=,BMPART) KEYM "IX01" MASK="?"
  ```

  The ,=, following BMPQTY has the effect of placing all question marks in that field. If you leave out the ,=, the value in the numeric field is used instead.

- All key matching is case-sensitive.

Examples

Use KEYF to retrieve all records, in ascending order, along an index composed of a single text key, BMPART. This example can be used in DIRECTORY style programs where a REPOINT value may be specified to reposition the display at a certain point in the sequence.

```
BMPART=REPOINT
GET REC(BMPART) KEYF “IX01” LEVEL=1
I=1
WHILE I<16 AND (REC=”NEXT” OR REC=”FOUND”)
 I=I+1
 GET REC KEYF “IX01” LEVEL=I
END
CONVERSE MAP
```

Use KEYM to retrieve all records with a ‘7’ in position 4 of the text key field BMPART. In this example, the index secondary key is composed of two fields, BMPQTY and BMPART.

```
BMPART=”***7**”
GET REC(BMPQTY,=,BMPART) KEYM “IX02” MASK=”*”
WHILE REC=”NEXT”
 SHOW BMPART
 GET REC KEYM “IX02”
END
```

Use KEYS to retrieve all records containing the string “gasket” in the BMDESC field. In this example, the index secondary key is composed of the single text field, BMDESC.

```
BMDESC=”gasket”
GET REC(BMDESC) KEYS “IX03” MASK=”*”
WHILE REC=”NEXT”
 SHOW BMDESC
 GET REC KEYS “IX03”
END
```

GET (RDM user view)

NOTE

This statement applies to SUPRA RDM users only.

```
                    ┌                         ┐
                    │          ┌ NEXT  ┐      │
                    │          │ PRIOR │      │
GET view - name     │ [(key,...)] FIRST │      │ [ENQUEUE] [LEVEL = level – number]
                    │          │ LAST  │      │
                    │          └       ┘      │
                    │ AT mark                 │
                    │ SAME                    │
                    └                         ┘
```

view-name

Description *Required*. Specifies the symbolic name of the RDM user view you want to access.

Format MANTIS symbolic name as defined in a previously executed VIEW statement

(*key*,...)

Description *Optional*. Specifies the key of the desired row in the view. Each key parameter must match the data type of the corresponding KEY field in the RDM user view. If trailing key fields are omitted, the resulting partial key is used to access the view.

Format Expression of the same type and size as the corresponding key field in the user view

AT *mark*

Description *Optional*. Repositions to the row previously marked with the MARK statement.

Format Four-character text expression with a value obtained from a previously executed MARK statement

SAME

Description *Optional*. Retrieves the same row previously accessed.

Consideration If no current row exists, MANTIS returns a NOTFOUND status.

NEXT PRIOR FIRST LAST

Description *Optional*. Indicates the order of retrieval for rows.

Default NEXT

Options NEXT retrieves the next row in the user view with the specified keys. If you do not supply keys, MANTIS returns the next sequential row. If no current row exists, GET...NEXT operates as GET...FIRST.

PRIOR retrieves the previous row with the specified keys. If no current row exists, GET...PRIOR operates as GET...LAST. If you are positioned at the first record when you issue GET...PRIOR, MANTIS returns NOTFOUND. A subsequent GET...PRIOR returns the last record in the file. If you do not supply keys, MANTIS retrieves the row before the currently established position of a given key. However, after processing all prior rows for a key, MANTIS displays a message and halts execution. If TRAP is in effect, MANTIS returns ERROR.

FIRST retrieves the first row in the user view with the specified keys. If you do not supply keys, MANTIS returns the first row.

LAST retrieves the last row in the view for the specified keys. If you do not supply keys, MANTIS displays an error message and halts execution. If TRAP is in effect, MANTIS returns ERROR.

ENQUEUE

Description *Optional*. Tells MANTIS to lock out all other users' modifications to the row being retrieved until the row has been updated, deleted, or dequeued by a COMMIT statement.

LEVEL=*level-number*

Description	*Optional*. Specifies the level of MANTIS array elements where the fields in the row are copied.
Default	LEVEL=1
Format	Arithmetic expression which evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding VIEW statement
Consideration	You must specify LEVEL=*level-numbe*r if the corresponding VIEW statement specifies *levels* unless you want the default value.

General considerations

- Before you can read a row from an RDM user view, you must open it with a VIEW statement.
- The number of keys you specify in the GET statement must be less than or equal to the number of keys defined in the RDM user view.
- MANTIS treats any omitted keys as generic keys. The use of generic keys allows direct access to a specific row or sequential access to many rows. All occurrences of a particular unspecified key are returned as long as the other keys are satisfied.
- The order of the specified keys must correspond to the order of key fields in the view. This is the order of the key fields in the view, unless you used SELECT(*field-list*,...) on the VIEW statement to define a user view with a different field order. You cannot omit a key which occurs between two keys you want to specify. (For example, you cannot specify the first and third keys without specifying the second key.)
- You need not use ENQUEUE unless you intend to update the row with new values derived from current values. When you execute the UPDATE or DELETE statements, the automatic hold facility performs the lock prior to modifying the row.
- The current row for SAME, NEXT, and PRIOR is the row accessed by the most recent GET or INSERT (regardless of level).

- MANTIS copies the columns in the row to the corresponding MANTIS arrays and variables. You can obtain the status of the GET by using the "view-name" built-in text function. The values are FOUND, NOTFOUND, and DATA. MANTIS may return a status of ERROR or HELD if TRAP is in effect.

  FOUND MANTIS successfully retrieved the row identified by the supplied key.

  NOTFOUND The row identified by the supplied key does not exist.

  DATA A field which should be numeric is not.

  ERROR An error occurred while accessing the user view. Something may be wrong with the database or you may have tried to perform an invalid function on the user view.

  HELD MANTIS failed to retrieve the row because it was held by another user. Note, in IBM compatibility mode, HELD is not returned, and ERROR is returned for this condition.

- Further status information may be obtained using the text functions FSI, ASI, and VSI. FSI indicates the success or failure of the GET function. ASI indicates the status of each field in the row. VSI indicates the highest field status within the row. For a complete discussion of these status functions, see "DBCS support feature" on page 455.

- If there is only one row for a given key and you try to use the same key with a keyed GET statement to access the row a second time, you receive a NOTFOUND status. This message indicates that there are no more occurrences of this particular row. Use a GET...SAME statement if you want to retrieve the same row again.

- If the underlying file system cannot perform the GET...PRIOR or GET...LAST functions, MANTIS displays a message and halts execution. If TRAP is in effect, MANTIS returns the ERROR status.

- A series of GET...NEXTs loops back to the first row and continues if you do not check for the NOTFOUND status.

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality has been disabled by the COMMIT OFF statement. Resources held by the GET... ENQUEUE would then be released on the next terminal input.

Example

```
10 VIEW CUSTOMER("CUST")
20 SCREEN MAP("CUST_UPDATE")
30 GET CUSTOMER FIRST
60 WHILE CUSTOMER="FOUND" AND MAP<>"CANCEL"
70 .CONVERSE MAP
80 .GET CUSTOMER
90 END
```

HEAD

Use the HEAD statement to center a heading on the top line of the screen and set it to high intensity.

HEAD *heading*

heading

Description *Required*. Specifies the heading to be displayed at the top of the screen.

Format Text expression

Considerations

- Headings do not appear when you CONVERSE a formatted screen, but only when MANTIS executes a SHOW statement for unformatted screen output. (You can also do a SHOW t that goes on the message line of a formatted screen). Headings appear in scroll mode when MANTIS executes a SHOW, WAIT, or OBTAIN statement. Headings are not used for a CONVERSE or PROMPT.
- You can specify only one heading for each screen I/O. If more than one HEAD statement is executed, the most recent HEAD statement specifies the heading for a screen I/O.
- Specify HEAD “” to clear a previous heading because CLEAR does not affect the heading.
- MANTIS evaluates the *heading* at the execution of the HEAD statement and does not re-evaluate it at terminal output.
- See also “SHOW” on page 334.

Example The following example shows how the HEAD statement is used to center a heading on a screen:

```
 10 ENTRY BUZZ_PHRASE_GENERATOR
 20 .DO SET_UP_VOCABULARY
 30 .HEAD "BUZZ PHRASE GENERATOR"
 40 .CLEAR
 50 .SHOW "I WILL GENERATE A SCREEN FULL OF BUZZ"
 55 .'"PHRASES EVERY TIME YOU PRESS 'RETURN'. "
 60 .'"WHEN YOU WANT TO STOP, ENTER 'CANCEL'."
 70 .UNTIL KEY= "CANCEL"
 80 ..INDEX=1
 90 ..UNTIL INDEX=22
100 ...A=INT(RND(10)+1)
110 ...B=INT(RND(10)+1)
120 ...C=INT(RND(10)+1)
130 ...SHOW FIRST(A)+""+SECOND(B)+""+NOUN(C)
140 ...INDEX=INDEX+1
150 ..END
160 ..WAIT
170 .END
180 .CHAIN "GAMES_MENU"
190 EXIT
```

IF-ELSE-END

Use the IF-ELSE-END statement to execute one block of statements if a specified condition is true, another block if the condition is false. The ELSE portion of this statement is optional.

IF *expression*

    [*true-block*]

[ELSE

    ***false-block*]**

END

expression

Description	*Required*. Specifies the condition that determines whether true-block or false-block is executed.
Format	Relational or arithmetic expression that evaluates to TRUE (nonzero) or FALSE (zero)

true-block

Description	*Optional*. Contains the block of statements you want to execute if expression is TRUE (non-zero).
Consideration	If expression is TRUE, MANTIS executes the statements in *true-block* (if present) and resumes at the statement following END.

false-block

Description *Optional*. Contains the block of statements you want to execute if expression is FALSE (zero).

Consideration If expression is FALSE, MANTIS executes the statements in *false-block* (if present) and resumes at the statement following END.

General consideration

- See also “WHEN-END” on page 399.

Example

```
 70 WHILE ONE<>TWO AND ONE>ZERO AND TWO>ZERO
 80 .IF ONE<TWO
 90 ..ONE=ONE-INT(ONE/TWO)*TWO
100 .ELSE
110 ..TWO=TWO-INT(TWO/ONE)*ONE
120 .END
130 END
```

INSERT

Use the INSERT statement to insert a new record into a MANTIS file, an external file, an ULTRA file, or a row into an RDM user view.

INSERT (MANTIS file)

INSERT *file-name* [LEVEL=*level-number*]

file-name

Description	*Required*. Specifies the symbolic name of the MANTIS file in which you want to insert a record.
Format	MANTIS symbolic name as defined in a previously executed FILE statement

LEVEL=*level-number*

Description	*Optional*. Specifies which level of MANTIS array elements contains the fields in the record you want to insert.
Default	LEVEL=1
Format	Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding FILE statement
Consideration	You must specify LEVEL=*level-number* if the corresponding FILE statement specifies *levels* unless you want the default value.

General considerations

- ♦ You must execute a corresponding FILE statement before you can execute the INSERT statement.
- ♦ The contents of key data elements identify where the record is to be inserted.

- ♦ You can obtain the status of the INSERT by using the “file-name” built-in text function. The status is a null text value if the INSERT was successful. If the INSERT is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the system error message that may have been returned. This information is also available to MANTIS fault handling routines (see the SET TRAP statement). MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

  LOCK The password specified in the FILE statement for this file profile is not valid for deletions or insertions.

  ERROR A physical error occurred while inserting the record, the file was full or the file contained a record with the same key as the record to be inserted.

- ♦ MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

- ♦ **OpenVMS** If RMS Journaling is active and the file is RU Journal , the following occurs:

  - The START_RU (Recovery Unit) occurs.

  - The record is locked until COMMIT or RESET.

  - You may back out the insert via RESET.

- ♦ **OpenVMS** During the START_RU, if RMS Journaling is active, all RU Journal led files opened for update (including the MANTIS File) automatically join the Recovery Unit. This implies that a release of any of these files is delayed until the next COMMIT or RESET.

- ♦ **OpenVMS** If RMS Journaling is not active, and the file is RU Journal led, MANTIS issues a fault.

Example

```
 10 ENTRY INDEX
 20 .FILE RECORD("INDEX","SERENDIPITY")
 30 .SCREEN MAP("INDEX")
 40 .GET RECORD
 50 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
 60 ..CONVERSE MAP
 70 ..WHEN MAP="PF1"
 80 ...INSERT RECORD
 90 ..WHEN MAP="PF2"
100 ...DELETE RECORD
110 ..WHEN MAP="PF3"
120 ...UPDATE RECORD
130 ..END
140 ..GET RECORD
150 .END
160 EXIT
```

INSERT (external file)

INSERT *access-name* [LEVEL=*level-number*]

access-name

Description	*Required*. Specifies the symbolic name of the external file in which you want to insert a record.
Format	MANTIS symbolic name as defined in a previously executed ACCESS statement
Consideration	If the file has been released, it is automatically reopened, but the position in the file is lost. For example, if a GET...NEXT is subsequently executed, the first record in the file is returned regardless of which record was being accessed before the release.

LEVEL=*level-number*

Description	*Optional*. Specifies which level of MANTIS array elements contains the fields in the record you want to insert.
Default	LEVEL=1
Format	Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ACCESS statement
Consideration	You must specify LEVEL=*level-number* if the corresponding ACCESS statement specifies *levels* unless you want the default value.

General considerations

- You must execute a corresponding ACCESS statement before you can execute the INSERT statement.
- For indexed files, the contents of key data elements identify where the record is to be inserted.
- For sequential files, MANTIS inserts the record at the end of the file.
- For relative files, the Relative Record Number (RRN) contained in the associated reference variable (defined during External File View Design) identifies the record to be inserted.

- You can obtain the status of the INSERT by using the “access-name” built-in text function. The status is a null text value if the INSERT was successful. If the INSERT is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the system error message that may have been returned. This information is also available to MANTIS fault handling routines (see the SET TRAP statement). MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

  LOCK The password specified in the ACCESS statement for this file view is not valid for deletions or insertions.

  ERROR A physical error occurred while inserting the record, the file was full or an indexed or relative file contained a record with the same key or RRN as the record to be inserted.

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

- **OpenVMS** If RMS Journaling is active and the file is RU Journal led, the following occurs:

  - The START_RU (Recovery Unit) occurs.
  - The record is locked until COMMIT or RESET.
  - You may back out the insert via RESET.

- **OpenVMS** During the START_RU, if RMS Journaling is active, all RU Journal led files opened for update (including the MANTIS File) automatically join the Recovery Unit. This implies that a release of any of these files is delayed until the next COMMIT or RESET.

- **OpenVMS** If RMS Journaling is not active, and the file is RU Journal led, MANTIS issues a fault.

Example

```
 20 .ACCESS RECORD("INDEX","SERENDIPITY",16)
 30 .SCREEN MAP("INDEX")
 40 .CONVERSE MAP
 50 .COUNTER=1
 60 .WHILE MAP<>"CANCEL" AND COUNTER<17
 70 ..WHEN INDICATOR(COUNTER)="G"
 80 ...GET RECORD LEVEL=COUNTER
 90 ..WHEN INDICATOR(COUNTER)="D"
100 ...DELETE RECORD LEVEL=COUNTER
110 ..WHEN INDICATOR(COUNTER)="I"
120 ...INSERT RECORD LEVEL=COUNTER
  .
  .
  .
```

INSERT (ULTRA file view)

NOTE

This statement applies to SUPRA PDM users only.

$$\textbf{INSERT}\ \textit{ultra-name}\ \begin{bmatrix}\textbf{FIRST}\\ \underline{\textbf{LAST}}\\ \textbf{BEFORE}\ \textit{refer}\\ \textbf{AFTER}\ \textit{refer}\end{bmatrix}\ \textbf{[LEVEL} = \textit{level-number}\,\textbf{]}$$

ultra-name

Description *Required*. Specifies the symbolic name of the ULTRA file in which you want to insert a record.

Format MANTIS symbolic name as defined in a previously executed ULTRA statement

FIRST
LAST
BEFORE
AFTER

Description *Optional*. Specifies the relative position within a set (chain) of records at which you want to insert a record.

Default LAST

Options

FIRST	inserts the record as the first record of the set
LAST	inserts the last record in the set
BEFORE	inserts the record before the one specified by refer
AFTER	inserts the record after the one specified by refer

Consideration These parameters are only applicable to related files.

refer

Description	*Optional*. Identifies the position in a related file before or after which you want to insert a record.
Format	Four-character text expression
Consideration	The refer value is usually the value of the reference variable.

LEVEL=*level-number*

Description	*Optional*. Specifies which level of MANTIS array elements contains the fields in the record you want to insert.
Default	LEVEL=1
Format	Arithmetic expression which evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ULTRA statement
Consideration	You must specify LEVEL=*level-number* if the corresponding ULTRA statement specifies *levels* unless you want the default value.

General considerations

- ♦ You must execute a corresponding ULTRA statement before you can execute the INSERT statement.
- ♦ The current position is returned in the reference variable specified during ULTRA File View Design. For more details, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300. You may save and restore this position using a text variable of at least four characters in length.
- ♦ You can obtain the status of the INSERT by using the “ultra-name” built-in text function. The values are GOOD and HELD.

GOOD	MANTIS successfully inserted the record.
HELD	MANTIS failed to insert the record because an associated record was held by another user.

- MANTIS may also return a status of DUPLICATE, LOCK, NOTFOUND, or ERROR if TRAP is in effect.

DUPLICATE	A record with the same key already exists.
LOCK	The password specified in the ULTRA statement is invalid.
NOTFOUND	The related file chain set with the requested key does not exist.
ERROR	An error occurred while inserting the record.

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

Example

```
 20 ULTRA BILL(“BOM”,”ASSEMBLY”,8)
  .
  .
  .
100 INSERT BILL BEFORE BILL_REFER(COUNT-1)LEVEL=COUNT
```

INSERT (RDM user view)

NOTE

This statement applies to SUPRA RDM users only.

$$\textbf{INSERT}\ \textit{view-name}\ \begin{bmatrix}\underline{\textbf{NEXT}}\\ \textbf{PRIOR}\\ \textbf{FIRST}\\ \textbf{LAST}\end{bmatrix}\left[\textbf{LEVEL} = \textit{level-number}\right]$$

view-name

Description	*Required*. Specifies the symbolic name of the RDM user view in which you want to insert a row.
Format	MANTIS symbolic name as defined in a previously executed VIEW statement

<u>NEXT</u>
PRIOR
FIRST
LAST

Description	*Optional*. Specifies where the row is to be inserted relative to the current position.	
Default	NEXT	
Options	NEXT	places the row in the RDM user view following the current row provided the keys are the same. If the keys are different or if the current position has not been established, INSERT...NEXT operates as INSERT...LAST.
	PRIOR	places the row in the RDM user view before the current row providing the keys are the same. If the keys are different, or if no current row has been established, INSERT...PRIOR operates as INSERT...FIRST.
	FIRST	places the row in the RDM user view so that a subsequent GET...FIRST using the same key values retrieves this row.
	LAST	places the row in the RDM user view so that a subsequent GET...LAST using the same key values retrieves this row.

LEVEL=*level-number*

Description	*Optional*. Specifies which level of MANTIS array elements contains the fields in the row you want to insert.
Default	LEVEL=1
Format	Arithmetic expression which evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding VIEW statement
Consideration	You must specify LEVEL=*level-number* if the corresponding VIEW statement specifies levels unless you want the default value.

General considerations

- ♦ You must execute a corresponding VIEW statement before you can execute the INSERT statement.
- ♦ The current row for NEXT and PRIOR is the row accessed by the most recent GET or INSERT (regardless of level).
- ♦ You can obtain the status of the INSERT by using the “view-name” built-in text function. The values are GOOD and HELD.

  GOOD MANTIS successfully inserted the row.

  HELD MANTIS failed to insert the row because an associated row was held by another user.

- ♦ MANTIS may also return a status of DUPLICATE or ERROR if TRAP is in effect.

  DUPLICATE A row with the same key already exists.

  ERROR An error occurred while inserting the row. Something may be wrong with the database or you may have tried to perform an invalid function on the user view.

- ♦ You may obtain further status information using the text functions FSI, ASI, and VSI. FSI indicates the success or failure of your command. ASI indicates the status of each field in the row. VSI indicates the highest field status within the row. For a complete discussion of these status functions, see “DBCS support feature” on page 455.

- ♦ If the row to be inserted is uniquely keyed and if the value of the key to be inserted already exists in the database, the INSERT fails. If TRAP is on, MANTIS returns DUPLICATE; otherwise, MANTIS displays an error message and halts execution.
- ♦ If your database administrator specified ordering in the view definition or if the physical DBMS does not allow program control of ordering, MANTIS ignores the specification on the INSERT statement.
- ♦ If TRAP is off for this user view and MANTIS receives a failure status from RDM, MANTIS issues a RESET to ensure database integrity. If TRAP is on and ERROR is returned and the MANTIS program does not perform the RESET, it is possible that MANTIS will do only part of the insertion.
- ♦ Your database administrator may disallow insertions. If so, MANTIS returns the ERROR status if TRAP is in effect. Check the FSI message for the reason. If TRAP is not in effect, MANTIS displays a message and halts execution.
- ♦ MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

Example

```
 10 VIEW CUSTOMER("CUST")
 20 SCREEN MAP("CUST_UPDATE")
 30 SHOW "ENTER CUSTOMER NUMBER:"
 40 OBTAIN CUST_NO
 50 GET CUSTOMER(CUST_NO)
 60 IF CUSTOMER<>"FOUND"
 70 .INPUT_OK=FALSE
 80 .UNTIL INPUT_OK
 90 ..CONVERSE MAP
100 ..DO EDIT_INPUT
110 ..WHEN EDIT_OK
120 ...INPUT_OK=TRUE
130 ..END
140 .END
150 .INSERT CUSTOMER
160 .SHOW "CUSTOMER INFORMATION INSERTED"
170 ELSE
180 .SHOW "CUSTOMER ALREADY EXISTS"
190 END
```

INT

Use the INT function to return the integer value of an arithmetic expression, truncating fractions towards zero.

INT(*a*)

a

Description *Required*. Specifies the arithmetic expression whose integer you want returned.

Format Arithmetic expression

General consideration

- See also “LET” on page 273 and “RND” on page 321.

Examples

```
INT(45)  returns    45
INT(45.5)    returns    45
INT(-45.5)   returns   -45
INT(-.5) returns   (0)
```

INTEGER

Use the INTEGER statement to define integral numeric variables or arrays of integral numeric variables. INTEGER variables and array elements are signed 4-byte quantities with a domain of -2147483648 to 2147483647. Values are initially set to zero.

INTEGER *integer-name* [(*dimension*,...)],...

integer-name

Description *Required*. Define the symbolic name of the numeric variable or array.

Format A MANTIS symbolic name

Consideration No processing occurs if *integer-name* is already defined.

dimension

Description *Optional*. Specify an array dimension.

Format Arithmetic expression that evaluates to a value in the range 1–*max*, where *max* is the value of the MAXDIMSIZE MANTIS Option.

Considerations

- Use one dimension parameter to specify a one-dimensional array, two dimension parameters to specify a two-dimensional array, and so on.
- The maximum number of dimensions you can specify is determined by the value of the MAXNODIMS MANTIS Option. Each dimension specifies the maximum value of the corresponding array subscript.

General considerations

- The maximum number of variable names you can specify is determined by the value of the MAXVARS MANTIS Option.
- All binary operations involving only one INTEGER operand cause that operand to be converted to either BIG or DECIMAL, depending on whether the other operand type is SMALL/BIG or DECIMAL, respectively.
- See also “BIG” on page 144, “DECIMAL” on page 175 and “SMALL” on page 344.

- Auto mapping of variables takes place during complex variable declarations such as FILE and ACCESS statements. MANTIS generates a dissimilarity fault (212) if you attempt to map a new variable to an existing one with less precision. For example, if an ACCESS statement would define a DECIMAL variable with a precision of 10, and that variable is already defined in the work area with a precision of less than 10, a 212 fault will occur. For the purposes of dissimilarity fault checking, MANTIS data types are attributed with the following precision.

Type	Precision
SMALL	6
INTEGER	9
BIG	14
DECIMAL	1–31 as defined

So for example, it is illegal to auto-map an INTEGER to an existing SMALL variable, and a BIG or DECIMAL(10) variable cannot be mapped onto an existing INTEGER variable.

Example

```
 10 INTEGER I

 20 I = 1
   30 WHILE I < 10
     .

     .

     .
 90  I = I + 1

100 END
```

INTERFACE

Use the INTERFACE statement to define an interface your program accesses. MANTIS retrieves the interface profile from the library and defines MANTIS variables for each field in the interface profile.

INTERFACE *interface-name*(*libname,password*[,PREFIX][,*levels*]),...

interface-name

Description	*Required*. Specifies the symbolic name for the interface profile in subsequent CALL statements.
Format	Unique MANTIS symbolic name
Consideration	No processing occurs if *interface-name* is already defined.

libname

Description *Required*. Specifies the name of the interface profile as it was saved during Interface Design.

Format Text expression that evaluates to a library name in the format:

```
[user-name:]interface-name
```

Consideration If the interface profile is in your own library, you do not need to specify *user-name*.

password

Description	*Required*. Specifies the password needed to access the interface.
Format	Text expression that evaluates to the password specified during Interface Design

PREFIX

Description *Optional*. Indicates that MANTIS place the prefix "*interface-name*_" on all variable names associated with this interface. If, for example, the interface BOTTLES has a field named VOLUME, the following statement causes the corresponding variable BIN_VOLUME to be defined in the work area:

```
10 INTERFACE BIN("BOTTLES","MAGIC",PREFIX)
```

levels

Description *Optional*. Specifies the number of levels for this interface that you want to use in CALL statements.

Format Arithmetic expression that evaluates to a value in the range 1–255

Considerations

- If you specify *levels*, you must specify LEVEL=*level-number* (unless you want the default LEVEL=1) in all CALL statements for the interface.
- If you do not specify *levels*, MANTIS defines an INTEGER, SMALL, BIG, DECIMAL or TEXT variable or array for each field in the interface profile according to the type and dimensions specified during Interface Design.
- If you specify *levels*, MANTIS adds an extra dimension to each field to allow a separate value to be stored at each level. MANTIS defines a one-dimensional array instead of a variable, a two-dimensional array instead of a one-dimensional array, and so on. The first dimension of each array is *levels*.

General considerations

- Consult your Master User before using the INTERFACE statement.
- The INTERFACE and FILE statements have the same format.
- See also “CALL” on page 147.

Example

```
20 INTERFACE MASTER("CUSTOMERS","ALIBABA",10)
30
40
50
60 CALL MASTER("GET",1234)LEVEL=2
```

interface-name

Use the interface-name function to return the eight-character interface status which may be set by the interface program.

interface-name

interface-name

Description *Required*. Specifies the symbolic name of the interface program.

Format MANTIS symbolic name as defined in a previously executed INTERFACE statement

Example

```
20 INTERFACE MASTER("CUSTOMERS","ALIBABA",10)
30
40
50
60 CALL MASTER("GET",1234)LEVEL=2
SHOW MASTER
```

KEY

Use the KEY function to return a text value identifying your response to the most recent CONVERSE, OBTAIN, PROMPT, or WAIT statement. Also returns the value of the key pressed in response to the last MORE prompt issued as a result of the SHOW statement filling the screen.

KEY

General considerations

- KEY values are program function (logical key) names if returned by the key pressed, otherwise, the value entered in the Reply Field.
- For CONVERSE, OBTAIN, PROMPT, and WAIT: If the Reply Field is not empty, the key value is the value in the Reply Field. If the Reply Field is empty, the key value is the key sequence value.
- CLEAR sets the KEY to "CLEAR." CANCEL sets KEY to "CANCEL" so the program can detect it.
- The value remains available if you CHAIN to another program or DO an external subroutine.
- During field sensitive validation for CONVERSE, field entry and exit routines can use the KEY function to determine how a field gained or lost the input focus. For example, when you press the TAB key to move the cursor to the next field, the value of KEY is set to "TABFLD."
- See also "screen-name" on page 324.

Example

```
 10 ENTRY BUZZ_PHRASE_GENERATOR
 20 .DO SET_UP_VOCABULARY
 30 .HEAD "BUZZ PHRASE GENERATOR"
 40 .CLEAR
 50 SHOW "I WILL GENERATE A SCREEN FULL OF BUZZ"
 60 .'"PHRASES EVERY TIME YOU PRESS 'RETURN'."
 70 .'"WHEN YOU WANT TO STOP, ENTER 'CANCEL'."
 80 ..UNTIL KEY="CANCEL"
 90 ..INDEX=1
100 ..UNTIL INDEX=22
110 ...A=INT(RND(10)+1)
120 ...B=INT(RND(10)+1)
130 ...C=INT(RND(10)+1)
140 ...SHOW FIRST(A)=""=SECOND(B)=""+NOUN(C)
150 ...INDEX=INDEX+1
160 ..END
170 ..WAIT
180 .END
190 .CHAIN "GAMES_MENU"
200 EXIT
```

LANGUAGE (function)

Use the LANGUAGE function to return the currently assigned language of the MANTIS user.

LANGUAGE

General considerations

- Your default language is set by your Master User in your User Profile.
- The user’s language can be altered by the LANGUAGE statement.

Example

```
SHOW LANGUAGE
ENGLISH
```

LANGUAGE (statement)

Use the LANGUAGE statement to change the current language assignment. The screen and prompter statements attempt to locate designs with a language code corresponding to your current default language. If a design for a given language cannot be found, the English language version is used if it exists.

LANGUAGE=*name*

name

Description *Required*. Specifies the language name you want to become the current language.

Format Text expression of 1–16 characters

General considerations

- ♦ Your initial default language is set by your Master User in your User Profile.
- ♦ The specified language name must exist in the Languages file maintained by the MASTER user. MANTIS ignores an invalid language specification.

Example

```
LANGUAGE="FRENCH"
SHOW LANGUAGE
FRENCH
```

LET

Use the LET statement to assign the value of an expression to a variable or array element or text substring.

[LET] ***variable*** **[ROUNDED[(*places*)]]=*expression*,...**

variable

Description	*Required*. Specifies the variable or array element or text substring to which you want to assign a value.
Format	Variable name or array element reference or text substring reference

Considerations

- A scalar text variable may be subscripted to identify the starting and optionally ending character positions of a substring within the string.
- A text array element reference may have optional additional substring subscripts following the last array element subscript.

ROUNDED(*n*)

Description	*Optional*. Specifies the number of decimal digits (*n*) to which you want the number carried out during the calculation (and before the number is assigned to a variable).
Format	Integer from 0–6, inclusive

Considerations

- ♦ Used when performing calculations whose results are real numbers.
- ♦ If ROUNDED is not used, two variables that seem to be equal (when displayed by SHOW statements) can actually compare unequal due to internal floating-point representation.
- ♦ The reserved word LET is optional.
- ♦ If you have not provided an explicit definition of the variable, v, MANTIS assigns a BIG type.
- ♦ Use the ROUNDED option when you want to control fractional results, for example, in currency calculations.
- ♦ You may also use CLEAR to set all elements of the variable *v*, to ZERO or NULL.
- ♦ See also “INT” on page 263.

places

Description	*Optional*. Specifies the number of decimal places to round an arithmetic expression before MANTIS assigns it to the numeric variable or array element.
Default	0
Format	Integer in the range 0–6
Consideration	If you specify ROUNDED without specifying places, MANTIS rounds to the nearest integer.

expression

Description *Required*. Specifies the value to be assigned to the variable or array element or text substring.

Format Arithmetic expression for assignment to an INTEGER, SMALL, BIG or DECIMAL variable or array element; text expression for assignment to a TEXT variable, array element, or substring

Consideration If you specify more than one expression, the variable must be a subscripted array reference. MANTIS increments the array subscript(s) after assigning each value. The right-most subscript is incremented first. Whenever a subscript exceeds its dimension, it is set to 1 and the next subscript to the left is incremented. MANTIS only uses the integer portion of numbers you enter for subscripts.

General considerations

- The keyword LET is optional.
- If the symbolic name has not already been defined either explicitly in an INTEGER, SMALL, BIG, DECIMAL or TEXT statement, or implicitly via an ACCESS, SCREEN, FILE, INTERFACE, ULTRA or VIEW statement, MANTIS automatically creates the variable as a BIG scalar variable.
- Use the ROUNDED option when you want to control fractional results, for example, in currency calculations.
- To assign a numeric value to a text variable or an array element, you must convert it to a text value using a built-in function (TXT or FORMAT—see the examples below).
- To assign a text value to a numeric variable or array element, you must convert it to a numeric value using the built-in function VALUE (as in the example below).

- A new text value can be assigned to a text substring by specifying one or two subscripts (see “Text substrings” on page 45):

    - If the first and last character positions are specified, the new value is truncated or padded with spaces to the same length as the substring. The length of the text variable remains unchanged unless the substring extends past the end of the old value.

    - If only the first character position is specified, the last character position is determined by the maximum length of the text variable and the new value is truncated to the corresponding length. The length of the text variable depends on the length of the new value.

  In either case, any positions between the end of the old text value and the start of the substring are padded with spaces.

Examples

```
10 LET ANSWER ROUNDED(2)=CAPITAL*(1+RATE/100)**LENGTH
```

is equivalent to

```
10 ANSWER ROUNDED(2)=CAPITAL*(1+RATE/100)**LENGTH
```

```
10 LET PRINTER=expression
```

is equivalent to

```
10 PRINTER=expression
```

```
10 A=(B+C)*D
20 BIG X
30 TEXT Y
40 LET X=VALUE(Y)
50 LET Y=TXT(X)
60 X=X+1
```

```
10 BIG PRIMES(10)
20 PRIMES(1)=2,3,5,7,11,13,17,19,23,29
```

```
10 TEXT ANSWER(20)
20 ANSWER(1,13)="The answer is"
30 ANSWER(17)=TXT(42)
```

```
10 TEXT LINE(80)
20 LINE = FORMAT(42, "The answer is Z####")
30 SHOW LINE
Results in:
The answer is 00042
```

MANTIS allows you to place logical expressions on the right side of a LET statement, so:

```
STATUS=(FIELD=SPACES(1,SIZE(FIELD)))
```

is equivalent to:

```
.STATUS=FALSE
.COUNT=SIZE(FIELD)
.WHEN FIELD=SPACES(1,COUNT)
..STATUS=TRUE
.END
```

LOG

Use the LOG function to return the natural logarithm of "*a*."

LOG(*a*)

a

Description *Required*. Specifies the arithmetic expression whose logarithm you want returned.

Format Arithmetic expression

Considerations

- ♦ The expression result is coerced to BIG. Decimal expression results may therefore lose some precision or may generate an arithmetic fault (310) if the magnitude is too large for a BIG data type.
- ♦ See also "E" on page 202 and "EXP" on page 208.

Example

```
10 X=1000
20 SHOW LOG(X)
RUN
6.90775527898
```

$LOGICAL

Use the $LOGICAL function to allow access to logical names from a MANTIS program.

$LOGICAL(*logical-name*[,*table-name*])

logical-name

Description	*Required*. Specifies a logical name.
Format	Text expression

table-name

Restriction	OpenVMS environments only.
Description	*Optional*. Specifies a logical name table.
Format	Text expression
Default	LNM$PROCESS

Examples

```
SHOW $LOGICAL(CUST)
SHOW $LOGICAL(CUST,CUST_TABLE)
SHOW $LOGICAL("FIND_THIS_NAME","LOOK_IN_THIS_TABLE")
```

LOWERCASE

Use the LOWERCASE function to return the specified text value with any uppercase letters changed to lowercase.

LOWERCASE(*t*)

t

Description *Required*. Specifies the text value to be changed to lowercase.

Format A text expression

General consideration

- ♦ See also "Text considerations" on page 43, "Storing text data" on page 43, "UPPERCASE" on page 387, ATTRIBUTE(TERMINAL)="NLS(*xxx*)" under "ATTRIBUTE (statement)" on page 111, and "TERMINAL" on page 353.

Example The following example shows how the LOWERCASE function is used to convert a text string to lowercase:

```
00110 SHOW LOWERCASE("abc $ ABC"):WAIT
     RUN
abc $ abc
```

The following example can be used to show how a case insensitive compare can be done on two text fields, A and B.

```
TEXT A,B
…
IF LOWERCASE(A)=LOWERCASE(B)
…(block of code for comparison equal)
END
```

NOTE

Lowercase translation can be modified by your System Administrator for your native language.

Example

```
SHOW LOWERCASE("Lend Me Your Ears")
lend me your ears
```

MARK

Use the MARK statement to save the current position in an RDM user view as established by the most recent GET, UPDATE, or INSERT statement. Before you can mark a user view, you must open it with a VIEW statement.

NOTE

This statement applies to SUPRA RDM users only.

MARK *view-name* AT *mark-name* [LEVEL=*level-number*]

view-name

Description	*Required*. Specifies the symbolic name of the RDM user view you want to mark.
Format	MANTIS symbolic name as defined in a previously executed VIEW statement

AT *mark-name*

Description	*Required*. Specifies where MANTIS should save the MARK information.
Format	MANTIS symbolic name
Consideration	Must be a text variable of length at least 4.

LEVEL=*level-number*

Description	*Optional*. Specifies the level number of the row that is marked.
Default	LEVEL=1
Format	Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding VIEW statement
Consideration	If the corresponding VIEW statement specifies *levels*, you must specify LEVEL=*level-number* unless you want the default value.

General considerations

- ALL MARKs on the RDM user view are invalidated by the RELEASE statement.
- A MARK text variable contains internal information which is not meaningful if displayed by functions such as SHOW.
- Use GET...AT mark-name to reset the current position in the view to the position saved by the MARK statement.
- There is no limit on the number of MARKs which can be saved.

Example

```
10 VIEW CUSTOMER("CUSTOMER-ACCOUNTS")
15 TEXT CUST_MARK(4)
20 GET CUSTOMER("R14148")
30 MARK CUSTOMER AT CUST_MARK
40 SHOW CUSTOMER_NUMBER
50 GET CUSTOMER("X00070")
60 SHOW CUSTOMER_NUMBER
70 GET CUSTOMER AT CUST_MARK
80 SHOW CUSTOMER_NUMBER
```

MODIFIED

Use the MODIFIED function to determine how many fields were modified or which fields, if any, on a screen were modified during the last CONVERSE statement. Because zero evaluates to FALSE, you can also use MODIFIED as a logical or arithmetic function.

$$\textbf{MODIFIED}\left(\ \mathit{screen-name}\begin{bmatrix} ,\mathit{field-name} \\ ,(\mathit{lrow},\mathit{lcol}) \end{bmatrix} \right)$$

screen-name

Description *Required*. Specifies the symbolic name of the screen you are testing.

Format MANTIS symbolic name as specified in a previously executed SCREEN statement

field-name

Description *Optional*. Specifies the symbolic name of the field you are testing.

Format Valid symbolic name of a field in the screen as specified during Screen Design

Considerations

- If you supply a field name, MANTIS returns TRUE (1) if you altered the specified field. Otherwise, MANTIS returns FALSE (0).
- If a substring is specified, the subscripts are ignored and the whole field is used.
- If you do not supply a field name, MANTIS returns the number of fields altered during the last physical I/O of the screen to the terminal. MANTIS returns 0 (FALSE) if no fields were modified. (NOTE: MANTIS evaluates 0 as FALSE and any other number as TRUE when the MODIFIED function is used in a conditional expression.)

(lrow,lcol)

Description *Optional*. Specifies the coordinates of the field you are testing. These are the row and column coordinates of the field within the logical display. The selected field is the one in the specified screen that contains these coordinates.

Format *lrow* and *lcol* must each be a numeric expression that evaluates to a value from 1 to 32767

General considerations

- MANTIS evaluates to a TRUE condition any non-zero value .
- MANTIS resets modified field indicators only when you converse a screen for physical I/O. In other words, MANTIS does not reset field indicators when you converse a screen with the WAIT option. If you add the UPDATE option to a CONVERSE statement, MANTIS allows you to modify all unprotected fields in a map set.
- The KMM (KEEP MAP MODIFIED) attribute prevents MANTIS from clearing the modified data tags of the specified screen. MANTIS ordinarily clears the modified data tags of all maps in the mapset for a CONVERSE UPDATE or for the active map in the case of a CONVERSE SET. If the modified data tags were cleared, a previously modified map returns FALSE for the MODIFIED function. If a map has the attribute KMM, then once modified, the MODIFIED function returns TRUE.

Examples

```
 10 SCREEN CLIENT_INFO("CLIENT_INFORMATION")
 20 CONVERSE CLIENT_INFO
 30 IF MODIFIED(CLIENT_INFO)
 40 .DO CLIENT PROCESSING: | ONE OR MORE FIELDS CHANGED
 50 END
    .
    .
    .
290 ENTRY CLIENT_PROCESSING
300 .FIELDS_TO_CHECK=MODIFIED(CLIENT_INFO)
310 .I=1
320 .LIMIT=SIZE(CLIENT_NAME,1)
330 WHILE I<=LIMIT AND FIELDS_TO_CHECK
340 ..IF MODIFIED(CLIENT_INFO,CLIENT_NAME(I))
350 ...| perform any edit checks
360 ...FIELDS_TO_CHECK=FIELDS_TO_CHECK-1
370 ..END
380 ..I=I+1
390 .END
400 EXIT
```

NEXT

Use the NEXT statement to proceed immediately to the next conditional repeat in a FOR-END, UNTIL-END, or WHILE-END statement or to the next WHEN condition in a WHEN-END statement.

NEXT

General considerations

- ♦ In FOR-END, UNTIL-END, and WHILE-END statements, NEXT results in the next repeat only if the respective condition is satisfied. If the condition is not satisfied, the statement after the END statement is executed next.
- ♦ In FOR-END statements, NEXT causes MANTIS to add the increment to the counter before comparing the counter with the final value.
- ♦ In WHEN-END statements, NEXT causes MANTIS to drop through to the next WHEN condition or to the END statement.
- ♦ In UNTIL-END, NEXT results in the repeat only if the until condition is not satisfied. If the until condition is satisfied, the statement after the END statement is executed.
- ♦ NEXT is not required before the END statement. Executing the END statement is sufficient to repeat the loop or next WHEN, based on condition checking.
- ♦ With nested logic statements, NEXT proceeds with the innermost enclosing FOR-END, UNTIL-END, WHEN-END, or WHILE-END statements where it occurs.
- ♦ See also “BREAK” on page 146, “RETURN” on page 320, “FOR-END” on page 214, “UNTIL-END” on page 373, “WHEN-END” on page 399 and “WHILE-END” on page 401.

Examples

```
 10 FOR L=1 TO MAXLINES:| For each screen line
 20 .IF NOT(MODIFIED(CUST_DETAILS,CUST_CREDIT(L)))
 30 . NEXT:| Continue if customer credit not modified
 40 .END
 50 .UPDATE CUSTOMER LEVEL=L:| Update customer record
 60 END
100 WHEN MODIFIED(SALE,DATE)
110 .DO CHECK_DATE(OK)
120 .IF OK
130 ..NEXT
140 .END
150 .ATTRIBUTE(SALE,DATE)="HIGHLIGHT"
160 WHEN MODIFIED(SALE,QUANTITY)
170 .DO CHECK_QUANTITY(OK)
180 .IF OK
190 ..NEXT
200 .END
210 .ATTRIBUTE(SALE,QUANTITY)="HIGHLIGHT"
220 END
230 CONVERSE SALE
```

NOT

Use the NOT function to return TRUE (1) for an arithmetic expression if "*a*" evaluates to FALSE (0); otherwise, NOT returns FALSE(0).

NOT(*a*)

a

Description *Required*. Specifies the arithmetic expression you want evaluated to true or false.

Format Arithmetic expression

General consideration

- ♦ See also "FALSE" on page 209, "SGN" on page 333 and "TRUE" on page 362.

Example

```
  10 IF NOT(OK AND I < LIMIT)
  .
  .
  .
 100 END
```

NULL

Use the NULL function to return a null (zero-length) text value ("").

NULL

General consideration

- ♦ The null string result should not be confused with "null values" or "missing values" associated with RDM user views. Use the ASI statement to set a "missing value" to an RDM user view prior to INSERT or UPDATE.

Example

```
A=NULL
```

NUMERIC

Use the NUMERIC function to determine if a text expression contains a valid number.

NUMERIC (*text-expression*)

text-expression

Description *Required*. Specifies the expression you want MANTIS to check.

Format Text expression

General considerations

- ♦ TRUE is returned if the text expression contains a valid number; FALSE is returned if it is not a valid number.
- ♦ A valid number is defined by the following rules:
  - May contain one decimal point*.
  - May contain commas in valid positions*.
  - May have a leading or trailing sign character, but not both. The sign character may be separated from the rest of the number by blanks.
  - Must contain at least one number (0–9) without embedded spaces.
  - May not contain currency notation ($, for example).
  - May not contain an exponent (scientific notation).

* Interpretation of the decimal point and comma is according to settings of the DECIMAL and THOUSANDS MANTIS Options, respectively.

Examples

Examples	Results
`NUMERIC ("123,456.789")`	`Returns True`
`NUMERIC ("$1234.55")`	`Returns False`
`NUMERIC ("-.05")`	`Returns True`
`NUMERIC ("")`	`Returns False`

`ABC="1234.55"`
`NUMERIC (ABC + "44")`
	`Returns True`

OBTAIN

Use the OBTAIN statement to obtain data values from an unformatted screen and assign them to numeric or text variables or array elements. You can also use OBTAIN to obtain unsolicited data entered on the bottom line of a formatted screen after the most recent CONVERSE statement was executed.

OBTAIN *variable*,...

variable

Description *Required*. Specifies a variable or array element for which you want to obtain a value.

Format MANTIS symbolic name or array element reference

General considerations

- You can mix different types of variables in an OBTAIN statement.
- When you enter values in response to an OBTAIN statement, separate the values with a semicolon (;). Note that you cannot enter a semicolon in a text variable unless it is the last variable in the list of variables being obtained.
- If there has been no intervening SHOW statement since the most recent CONVERSE statement, the OBTAIN statement obtains any data entered in the Unsolicited Input Field on the bottom line of the screen. If no data was entered in the Unsolicited Input Field, the variables in the OBTAIN statement remain unchanged.

- If the OBTAIN statement follows a SHOW, MANTIS waits for you to enter a line of data followed by RETURN. MANTIS assigns the values you enter to successive variables in the OBTAIN statement. If you enter more values than there are variables, the superfluous values are ignored. If you enter fewer values than there are variables, the variables for which there are no values remain unchanged.
- If there are data items pending from a previous SHOW statement ending with a comma (,) or a semicolon (;), the data items are displayed and input is accepted from the following position on the screen.

Examples

```
 5 FILE RECORD("DEMO_LIST","PASS3")
10 SCREEN MAP("DEMO")
20 SHOW "ENTER KEY FOR INQUIRY:";
30 OBTAIN ACCT_NUMBER
40 GET RECORD(ACCT_NUMBER)
50 WHILE MAP<>"CANCEL"
60 .CONVERSE MAP
70 .OBTAIN ACCT_NUMBER
80 .GET RECORD(ACCT_NUMBER)
90 END
 5 SHOW "PLEASE ENTER MONTH;DAY;YEAR"
10 OBTAIN MONTH,DAY,YEAR
```

During execution, this would appear on the screen as follows:

```
PLEASE ENTER MONTH;DAY;YEAR
4;2;84
```

$OPTION

Use the $OPTION function to return the value of the MANTIS option specified.

$OPTION(*option-name*)

option-name

Description *Required*. Specifies the name of the MANTIS option whose value you want returned.

Format Text expression

General consideration

- See also "SET $OPTION" on page 331.

Examples

```
SET $OPTION "MAXDIMSIZE=1024"
SHOW $OPTION("MAXDIMSIZE")
1024
```

OUTPUT

Use the OUTPUT statement to route output from the CONVERSE and SHOW statements and the LIST command.

$$\textbf{OUTPUT}\ \left\{\begin{array}{l}\textbf{SCREEN}\\ \textbf{PRINTER}\ [\textbf{VIA}\ \textit{exit}]\\ \textbf{SCREEN PRINTER}\ [\textbf{VIA}\ \textit{exit}]\end{array}\right\}$$

SCREEN

Description *Optional*. Indicates that MANTIS should send output from CONVERSE, SHOW, DISPLAY or LIST to your screen.

PRINTER

Description *Optional*. Indicates that MANTIS should send output from CONVERSE, SHOW, DISPLAY or LIST to the currently assigned printer.

Considerations

- You may set the system print command you wish to use on printing, by setting the PRINTCMD MANTIS Option.
- MANTIS is released with this option set to the appropriate command for your system.
- Your MANTIS user profile specifies the PRINTER-ID in terms of an external filename optionally followed by print command options. The PRINTER-ID is the default value given to PRINTER when you sign on to MANTIS.
- When MANTIS spawns a subprocess to perform a print command, any print command options specified by PRINTER are included in the print command issued to the system.
- The OUTPUT SCREEN statement closes any open print file.

SCREEN PRINTER*

Description *Optional*. Indicates that MANTIS should send output from CONVERSE, SHOW, DISPLAY or LIST to your screen and printer.

VIA *exit*

Description *Optional*. Tells MANTIS the name of a printer exit program. This parameter is for compatibility with MANTIS for the IBM mainframe and is not used by MANTIS.

Format Text expression

General considerations

- ♦ The OUTPUT statement clears all output lines from the SHOW statement that WAIT or OBTAIN have not forced out to the terminal.
- ♦ When OUTPUT PRINTER is in effect but OUTPUT SCREEN is not, the WAIT, OBTAIN and CONVERSE statements simulate the condition of the RETURN key being pressed (that is, the ENTER program function). MANTIS does not suspend the program.
- ♦ Removing MANTIS programs from memory closes any open print files.
- ♦ Different output is obtained in the Print File depending on whether you use OUTPUT PRINTER or OUTPUT SCREEN PRINTER. With OUTPUT PRINTER, the window size is the size of the print device (default (60,132)). With OUTPUT SCREEN PRINTER, the size of the terminal device (default (24,80)) is the size of the window. What you see on your physical screen is what is printed.
- ♦ In Program Design Programming Mode, the LIST command has a side effect of forcing output back to the screen if it was previously routed to the printer only.
- ♦ **OpenVMS** When output is directed to a print file only, the number of lines per page is calculated as 90% of the value returned by the OpenVMS run-time library routine, LIB$LP_LINES ( ).
- ♦ **UNIX** When output is directed to a print file only, the number of lines per page is 60 by default, but may be altered using the ATTRIBUTE(PRINTER)= statement.

Examples

```
OUTPUT SCREEN
```

Routes output to your screen. In programming mode, MANTIS automatically routes output to the screen unless you specify otherwise.

```
OUTPUT PRINTER
```

Routes output from CONVERSE, SHOW or DISPLAY statements (or the list command) to the currently assigned printer. You can print a listing of the current program by using the OUTPUT PRINTER statement in immediate mode followed by a LIST command. After MANTIS has printed the listing, output routing reverts to your screen.

```
OUTPUT SCREEN PRINTER
```

Routes output to both your screen and the printer.

PAD

Use the PAD statement to insert padding characters into a text variable, array element, or substring.

$$\textbf{PAD}\ \textit{string}\ [\textit{padding}] \begin{bmatrix}\textbf{BEFORE}\\ \underline{\textbf{AFTER}}\\ \textbf{ALL}\end{bmatrix}$$

string

Description *Required*. Specifies the text variable, array element, or substring which you want to pad with padding characters.

Format Name of a TEXT variable, the subscripted name of a TEXT array element or a reference to a TEXT substring

Consideration When the string is not a substring, the PAD statement sets the text variable or array element to its maximum length by inserting padding characters before or after the current text value. PAD has no effect if the current text value is already at the maximum length.

padding

Description *Optional*. Specifies the padding character.

Format Text expression

Default If you do not specify a padding character, MANTIS pads with spaces.

Consideration The first character is the padding character; any subsequent characters are ignored.

BEFORE

Description *Optional*. Inserts padding characters before the current characters in the string (that is, leading pad character).

AFTER

Description *Optional*. Inserts padding characters after the current characters in the string (that is, trailing pad character).

ALL

Description *Optional*. Inserts padding characters before and after the current characters in the string (that is, leading and trailing pad character).

Consideration If you specify ALL, MANTIS centers the current value within the padding characters.

General considerations

- ♦ If you do not specify BEFORE, AFTER, or ALL, MANTIS inserts padding characters after the current characters in the string.
- ♦ You cannot use BEFORE, AFTER, or ALL when padding a substring.
- ♦ See also "UNPAD" on page 369, "TEXT substrings" on page 354, and "LET" on page 273.
- ♦ You can pad a substring by specifying one or two subscripts (see "Text substrings" on page 45):
  - If the first and last character positions are specified, the length of the text variable remains unchanged unless the substring extends past the end of the old value.
  - If only the first character position is specified, the maximum length of the text variable determines the last position. The length of the text variable becomes the maximum length.

  In either case, the padding character is inserted in each position of the corresponding substring. Any positions between the end of the old text value and the start of the substring are padded with spaces.

Examples

The following examples of code and results show how the PAD statement can be used with various symbols on either side of an expression. Assume in the following examples, that TEXT A(20): this might be better as a table like the LET statement. General comment; This makes more sense to me for functions and statements like this than putting in code fragments with line numbers, SHOW and WAIT.

Statements following TEXT (A20) A=”ABC”	Result	Length
`PAD A "*" AFTER`	`ABC*****************`	20
`PAD A “-“ BEFORE`	`-----------------ABC` Can be used to right justify a field	20

`PAD A “+”`
`PAD A(6) "&"`
	`ABC++&&&&&&&&&&&&&&&`	20

`PAD A “+”`
`PAD A(7,11) "&"`
	`ABC+++&&&&&+++++++++`	20

`PAD A “-“ ALL`	`------ABC-------` Can be used to center a field	20

`TEXT B(20)`
`PAD B “*”`
	`********************`
	Can be used to initialize a field
		20

PASSWORD

Use the PASSWORD function to return a text value containing the password entered during MANTIS sign-on.

PASSWORD

Consideration If no password was used to sign on, PASSWORD returns a null string; otherwise, PASSWORD returns a text string of 1–16 characters. See also “USER” on page 387.

Example

```
10 ENTRY CUST_ENTRY
20 .SCREEN MAP("CUST_ENTRY")
30 .FILE REC("CUST_FILE",PASSWORD)
40 .FILE RECX("CUST_FILE",PASSWORD,PREFIX)
50 .CONVERSE MAP
60 .WHILE MAP<>"CANCEL"
  .
  .
  .
```

PERFORM

Use the PERFORM statement to invoke a system command. The command, which can run a user-written program, executes without accessing or interfering with the MANTIS work area.

PERFORM *command*

command

Description *Required*. Specifies the name of the system command to be invoked.

Format Text expression that evaluates to a system command

General considerations

- ♦ Standard input and standard output are available and are assigned to your terminal when your command is performed. Some of your commands, for example, "@," may change these assignments and you should assign them back to your terminal afterwards.
- ♦ A shorthand form of the PERFORM command is available. Entering a dollar sign ($) at the start of the line is equivalent to PERFORM. You can use the dollar sign ($) in place of the PERFORM command as a command in Program Design only, not in a program statement. The command after $ is a text value instead of a text expression, so that the following examples are equivalent.

```
$SHOW USERS
PERFORM "SHOW USERS"
```

- ♦ See "Programming techniques" on page 415 for additional information on the PERFORM statement as it works with external DO.
- ♦ There are performance considerations to be aware of when using subprocesses. See your Master User.

- ♦ If the command procedure writes to the screen it may interfere with optimized screen refreshing performed by subsequent MANTIS screen I/O statements. You have two ways of restoring the MANTIS screen appearance after a PERFORM statement.

  - Use the CLEAR statement in your program, following the PERFORM statement. You can decide if CLEAR is needed based on your knowledge of how specific command procedures behave.

  - Enable the AUTOREFRESH MANTIS Option, which unconditionally refreshes the full screen on the next screen I/O, regardless of whether the command procedure wrote to the screen.

  In either case the screen is not restored until the next screen I/O statement. CLEAR has a more immediate effect than AUTOREFRESH when the screen is operating in scroll mode and in full screen mode CLEAR erases the entire mapset whereas AUTOREFRESH has no effect on the mapset.

- ♦ **OpenVMS** The DELCHAIN, DELFACILITY, and DELIMMEDIATE MANTIS Options determine when a subprocess is terminated by MANTIS. Refer to *AD/Advantage MANTIS Administration OpenVMS/UNIX*, P39-1320, for more information.

- ♦ **OpenVMS** Subprocesses created by the PERFORM statement are not normally terminated after processing the command. You may use the DCL command ATTACH to attach to other subprocesses. New OpenVMS subprocesses are created if all the current subprocesses have images loaded.

- ♦ **OpenVMS** If you use the command string “SPECTRA,” it always attaches to the process that contains an image and was last used with the SPECTRA command.

- ♦ **OpenVMS** MANTIS supports a maximum of ten active processes. Avoid reaching the limit on the number of subprocesses, and avoid having all subprocesses with images (except for SPECTRA). If this occurs, MANTIS cannot attach to any subprocesses.

- ♦ **OpenVMS** Do not change the system message display format via the DCL SET MESSAGE command.

Example

```
10 ENTRY FACILITY
20 .SCREEN MAP("MENU")
30 .CONVERSE MAP
40 .WHILE MAP<>"CANCEL"
50 ..WHEN OPTION=1 OR MAP="PF1"
60 ...CHAIN "CUSTOMER_NAMES"
70 ..WHEN OPTION=2 OR MAP="PF2"
80 ...PERFORM "RUN INVOICES"
 .
 .
```

PI

Use the PI function to return the value of pi (3.14159265).

PI

General consideration

- See also “ATN” on page 110, “COS” on page 167, “SIN” on page 338, “TAN” on page 352, and “E” on page 202.

Example

This example shows how to compute the area of a circle.

```
10 SHOW "Enter Radius";
20 OBTAIN R
30 SHOW "Area of Circle ="; PI*R**2
```

The following example shows how convert between degrees and radians.

```
10 X=30
20 | COMPUTE SIN(X) where X is in degrees
30 DEGREES_TO_RADIANS=PI/180
40 Y=SIN(X*DEGREES_TO_RADIANS)
50 SHOW Y:WAIT
Results in .5
```

POINT

Use the POINT function to return the character position where addition or subtraction occurs in a text expression.

POINT $(t \pm t)$

t

Description *Required*. Specifies a text expression for which you want the addition or subtraction character positions returned.

Format Text expression

General consideration

- With subtraction (t1 – t2), Used to determine if and where one string exists within another. If *t2* does not exist in *t1*, zero is returned. Because 0 evaluates to FALSE, and any other value is true, POINT can be used in a logical expression (see example 3 – not numbered?). The value returned can also be used in substringing operations.

  For example, if you want those parts (PART_NAME) that contain the string “ECB” in their names, you could enter:

  ```
  IF POINT(PART_NAME-"ECB")
  ```

Statements	Results	Comments

`10 TEXT CUST_NO(11),DASH(1)`
`20 CUST_NO="222-22-2222":DASH="-"`
`30 A=POINT(CUST_NO-DASH`
	A=4	The (first) dash appears in the fourth position of CUST_NO

`10 .TEXT NAME(30)`
`20 .NAME="DOE,JOHN T."`
`30 .COMMA_PT=POINT(NAME-",")`
`40 .IF COMMA_PT`
`50 ..NAME=NAME(COMMA_PT+1)+" "+NAME(1,COMMA_PT-1)`
`60 .END`
	Name= “JOHN T. DOE”	Removes the first “.” (if any) from the string NAME and replaces with a space. Swaps the characters preceding and following the comma, if there is a comma. In line 50, COMMA_PT will be true if non-zero.

Statements	Results	Comments

`  10 NAME=”Doe,John”`
`2 20 FIRST_NAME=NAME(POINT(NAME-“,”)+1)`
`3 30 LAST_NAME=NAME(1,POINT(NAME-“,”)-1)`
	FIRST_NAME=”John”
	LAST_NAME=”Doe”
		Point value used to substringing indices. Assumes there is a comma at the appropriate point.

`10  IF NOT(POINT("PF1 PF2 PF3 PF4 PF5 ENTER CANCEL" - MAP))`
`20  .  MSG= "INVALID KEY PRESSED, MUST BE ENTER, CANCEL, or PF1 - PF5"`
`30  END`
	MSG is set if MAP does not contain one of the values in the quoted string.	Point used to validate “is contained in”.

`10  IF POINT (SHIP_DATE-"/"-"/"):| SEE IF SHIP_DATE HAS TWO "/"`
`  20  .MSG="ENTER SHIP DATE IN FORMAT DD/MM/YY"`
`  30  END`
	POINT returns FALSE (0) unless SHIP_DATE contains at least two “/” characters	POINT function shows how repeated + or - can be specified.
		
		This example does not check for three or more / or that there are valid dates between the /; additional edits are required for these checks.

NOTE

Parameters and arguments.

ARGUMENT is what the caller uses (i.e., the DO or CHAIN statements).

PARAMETER is what the callee uses (i.e., the ENTRY-EXIT statement.)

The ARGUMENT is “thrown” and the PARAMETER is “caught”. A mnemonic is alphabetical order, first comes the ARGUMENT then the PARAMETER.

PRINTER (function)

Use the PRINTER function to return a text value containing the current printer assignment.

PRINTER

Example

```
SHOW PRINTER
```

Display current printer assignment.

PRINTER (statement)

Use the PRINTER statement to direct MANTIS output to a print file or device.

PRINTER=*device*

device

Description *Required*. Specifies the name of the device to be assigned as the current printer or the name of a file to be produced. You can also specify parameters to the PRINT command to be issued when the file is closed.

Format Text expression that evaluates to the device name or logical name of a print file

General considerations

- If an OUTPUT statement specifies routing to a printer and the current program does not contain a PRINTER statement, MANTIS routes output using the default printer device name and spool command specified in your User Profile.
- You may set the system print command you wish to use on printing, by setting the PRINTCMD MANTIS Option.
- **OpenVMS** Routing output to a print file for later spooling is the preferred method under OpenVMS.

Examples

OpenVMS The following example:

```
PRINTER = "PRINT.LIS/DELETE/QUEUE = SYS$PRINT"
```

specifies that file PRINT.LIS should receive MANTIS output and that the file spools to the SYS$PRINT queue and deleted after printing.

OpenVMS The following examples are equivalent. They assign a physical device to receive printer output.

```
PRINTER = "TTAO"
10 TEXT DEVICE(4)
20 DEVICE = "TTAO"
30 PRINTER = DEVICE
```

UNIX The following example:

```
PRINTER = "print.lis"
```

specifies that the file print.lis should receive PRINTER output.

PROGFREE

Use the PROGFREE function to return the number of free bytes in the program work area.

PROGFREE

General consideration

- The PROGFREE function is not relevant to the MANTIS environment. The syntax is only supported for compatibility with MANTIS for the IBM mainframe.

Example

```
SHOW PROGFREE
```

PROGRAM

Use the PROGRAM statement to define an external subroutine to be invoked by a DO statement.

PROGRAM *program-name*(*libname*,*password*),...

program-name

Description *Required*. Defines the symbolic name of the external subroutine in subsequent DO statements.

Format Unique MANTIS symbolic name

libname

Description *Required*. Specifies the name of the program as you saved it in Program Design.

Format Text expression that evaluates to a library name in the format:

```
[user-name:]program-name
```

Considerations

- If the program is in your own library, you do not need to specify *user-name*.
- The *libname* parameter identifies a MANTIS program that must begin with a subroutine defined by ENTRY-EXIT statements. Only the first subroutine is accessible directly from the calling program, and any subsequent subroutines are internal to the external subroutine's program context.

password

Description *Required*. Specifies the password used when the program was saved during Program Design.

Format Text expression that evaluates to the password specified on the SAVE command (or the default password if none was specified)

Considerations

- ♦ If the program is in your own library, you do not need to specify the correct password.
- ♦ If you supply an incorrect password to a program in another user's library, you can still execute the external subroutine but you cannot list or edit it if it stops (because of a fault condition) in programming mode.
- ♦ Use the PROGRAM statement only when you are sure your program will execute an external subroutine.
- ♦ You must place an ENTRY-EXIT statement pair around the top-level routine in the subroutine you invoke. A program can DO both internal and external subroutines.
- ♦ To avoid overhead during re-execution, keep complex variable types (SCREEN, FILE, ACCESS, TOTAL, VIEW, and INTERFACE) in the highest-level routine possible and pass them as parameters.
- ♦ Do not make every routine external, but group related routines together in one program with multiple internal routines. You can use MANTIS COMPONENTS to keep a single version of source code in multiple programs.
- ♦ The *libraryn* and *program-namen* arguments for the PROGRAM statement are translated to uppercase upon execution of your program.

General considerations

- ♦ Statements such as SCREEN, FILE, ACCESS, ULTRA, VIEW, and INTERFACE which can generate a large amount of processing, should not be included in low-level external subroutines where they would be re-executed on each DO. If possible, insert them in a high-level routine and pass the associated symbolic names as arguments to low-level external subroutines.
- ♦ See "Programming techniques" on page 415 for additional information on the PROGRAM statement as it works with external DO.
- ♦ See also "CHAIN" on page 150 and "DO" on page 199.

Example

```
100 ENTRY EDIT_PROGRAM
 .
 .
 .
150 .PROGRAM EDIT_RTN("VALIDATION","COMMON")
160 .TYPE="CREDIT CHECK"
170 .DO EDIT_RTN(TYPE,CUST_NO,STATUS,MESSAGE)
180 .IF STATUS<>"GOOD"
190 ..DO ERROR_RTN(CUST_NO)
200 .END
210 .TYPE="SELECT SALES REP"
220 .DO EDIT_RTN(TYPE,CUST_NO,STATUS,MESSAGE)
230 .IF STATUS<>"GOOD"
240 ..DO ERROR_RTN(SALES_REP)
250 .ELSE
260 ..SALES_REP=MESSAGE
270 .END
280  EXIT
```

PROMPT

The PROMPT statement is used to display help information in scroll-mode. Help information can be obtained from MANTIS prompters or from external help libraries. MANTIS prompters allow a display-only mode of operation with a limit of 80 lines of help text. External help allows interactive help with an unlimited number of help lines.

PROMPT *helpkey* [, *helplib*]

helpkey

Description

Required. Specifies the key to locate the help information. This is either a standard MANTIS libname identifying a MANTIS prompter, or an external help key-path, which is one or more keywords separated by blanks.

Format

Text expression evaluating to a string in the format:

```
[user-name:]prompter-name
```

or

```
helpkey [helpkey]...
```

Considerations

- Press ENTER to view the next screen in the prompter. When prompter information ends, MANTIS returns control to the next line in the program. You can terminate the PROMPT statement by pressing CANCEL. You can terminate the program running the PROMPT statement by entering the KILL keyword in the lower left corner of the screen displaying the prompter.
- Because program comments consume space in a stored program and require run-time overhead, put large blocks of comments in a prompter. Use a PROMPT statement in immediate mode to view your comments
- *Prompter-names* and keypaths are limited to 30 characters.
- MANTIS prompters can be a maximum of 80 lines. You can chain prompters to expand their capacity (refer to Chapter 4 in *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300).

- ♦ If a MANTIS user name is specified in *helpkey*, the specified user library becomes the default in subsequent chained prompters, or until overridden by a user name in a chained prompter design.
- ♦ If the *helplib* parameter is omitted, MANTIS first interprets the *helpkey* as a MANTIS prompter libname. If the prompter is not found, then *helpkey* is assumed to be an external help key-path, and the default external help library is searched for the help information. In the latter case, the MANTIS user name, if specified, is stripped from the helpkey prior to invoking external help.
- ♦ MANTIS prompter chains can end by invoking external help if the last specified prompter to be chained to is not found in the MANTIS file.
- ♦ The MANTIS user profile used to obtain the default external help library defaults to the signed-on MANTIS user, or to any user name specified in helpkey or in the last chained prompter which specified a user name.
- ♦ If the *helplib* parameter is specified, MANTIS only interprets the *helpkey* as an external help key-path and does not attempt to locate the help information in the MANTIS File.
- ♦ **OpenVMS** If the MANTIS user profile does not specify a default external help library, then the normal external defaults apply, that is, HLP$LIBRARY, HLP$LIBRARY_1.

helplib

Description	*Optional*. Specifies an external help library to search for the helpkey.
Format	Text expression evaluating to a file specification.
Default	As specified in the MANTIS user profile where the user name defaults to the signed-on MANTIS user, or to any user name specified in *helpkey*.

Considerations

- ♦ This parameter is used for interactive help only and cannot be used to locate a MANTIS prompter in the MANTIS File.
- ♦ The specified helplib must exist, or an error is reported.

General considerations

- When an external help library is specified, the file specification (say XXXXXX) can be used to identify multiple help files for various languages. If the signed on user's language is LLLLLL, and XXXXXX_LLLLLL is a defined logical name, then it is used as the name of the help library; otherwise, XXXXXX is used.
- You can upgrade from MANTIS prompters to help without modifying your MANTIS programs by converting MANTIS prompters into a help library and using the prompter-names as level 1 keys in the help library. Do not specify a MANTIS user name in any help key, as MANTIS strips the user name from the helpkey parameter before invoking help.
- **OpenVMS** MANTIS invokes OpenVMS help by calling the OpenVMS help library routine LBR$OUTPUT_HELP( ). If the specified helpkey cannot be found in either the specified OpenVMS help library or one of the default OpenVMS help libraries, OpenVMS help prompts for a topic.
- **OpenVMS** MANTIS scroll-mode I/O routines are used by OpenVMS help for all I/O.
- **OpenVMS** You can use the CANCEL and KILL terminal functions to terminate both MANTIS prompter displays and interactive OpenVMS help. Either key terminates OpenVMS help immediately. For MANTIS prompters, CANCEL is used to terminate the display of each chained prompter, and KILL is used to terminate the PROMPT statement.

Examples

```
100 IF MAP="HELP"
110  PROMPT "MAIN_MENU"
120 END
200 IF MAP="HELP"
210  DO LOCATE_CURSOR(FIELD_NAME)
220  PROMPT SCREEN_NAME+" "+FIELD_NAME,"payroll"
230 END
```

RELEASE (function)

The RELEASE function returns a text string indicating the current release, environment, and copyright information about the MANTIS system that is executing.

RELEASE

Description Returns a text string with current release information

Example

The following example shows how to use the RELEASE function to return your current release level:

```
00010 SHOW RELEASE
```

Results in:

```
V2801.020.078.063 for HP9000/HPUX(UNIX)
```

RELEASE (statement)

Use the RELEASE statement to release internal storage used by an RDM user view, to close an external file or release internal storage used by an external subroutine.

$$\textbf{RELEASE}\begin{bmatrix} \textit{view} - \textit{name} \\ \textit{access} - \textit{name} \\ \textit{program} - \textit{name} \end{bmatrix}$$

view-name

Restriction SUPRA RDM users only.

Description *Optional*. Specifies the name of an RDM user view that you want to remove from internal storage.

Format MANTIS symbolic name as defined in a previously executed VIEW statement

Considerations

- MANTIS releases the internal storage used by the specified RDM user view. The current position in the view is then lost, and positions saved by MARK are no longer valid. If you execute a GET, UPDATE, INSERT, or DELETE on a released view, however, RDM automatically reopens that view. It is not necessary to re-execute the VIEW statement.
- MANTIS performs an implicit RELEASE of all open RDM views during MANTIS program context cleanup. This occurs in the following:
  - When the current test program context is released in Program Design as the result of NEW, LOAD, EDIT or RUN; or when the Program Design Facility is exited. Note that if the RUN command includes a line number, it is interpreted as a request to continue program execution, and no resources are released. In addition, this may cause unexpected results.
  - When a main program terminates or performs a CHAIN (without a DOLEVEL), unless the program is run from the Program Design Facility.

access-name

Description	*Optional*. Specifies the name of the external file you want to close.
Format	MANTIS symbolic name defined in a previously executed ACCESS statement
Consideration	MANTIS closes the specified external file so that your program does not prevent another program, such as an interface program, from opening it. The current position in the file is lost. If you execute a GET, UPDATE, INSERT, or DELETE on a closed external file, however, MANTIS automatically reopens it. It is not necessary to re-execute the ACCESS statement. If you do not use RELEASE to close an external file, it remains open until your program or subroutine terminates.

program-name

Description	*Optional*. Specifies the name of the external subroutine that you want to remove from internal storage.
Format	MANTIS symbolic name defined in a previously executed PROGRAM statement
Consideration	MANTIS removes the specified external subroutine from internal storage. After you execute an external subroutine for the first time, MANTIS keeps it in internal storage so that it does not need to be reloaded if you execute it again. MANTIS may run more efficiently if you use RELEASE to release the internal storage when the subroutine is no longer needed.

General considerations

- See “Programming techniques” on page 415 for additional information on the RELEASE statement and how it works with external DO.
- If you use RELEASE with no parameter, MANTIS releases the internal storage used by all RDM user views opened by VIEW statements.
- **OpenVMS** RELEASE of RU Journal led files is delayed until the next COMMIT or RESET if the recovery unit is active and it is opened for update.

Example

```
210 VIEW CUSTOMER("CUST")
220 .
230 .
240 RELEASE CUSTOMER
210 ACCESS RECORD("INDEX","SERENDIPITY",16)
220 .
230 .
240 .
250 RELEASE RECORD
310 PROGRAM EDIT_RTN("VALIDATION","COMMON")
320 .
330 .
340 DO EDIT_RTN(TYPE,CUST_NO,STATUS,MESSAGE)
350 .
360 .
370 RELEASE EDIT_RTN
```

RESET

Use the RESET statement to back out a Logical Unit of Work on an ULTRA, CIS_TM, and SUPRA database, on OpenVMS RMS RU journal led files and on external SQL databases. The contents of the database are RESET to the most recent COMMIT point. In the OpenVMS environment, RU Journal led file updates are backed out and delayed RELEASEs are performed. The RESET statement also unlocks all locked external file records.

RESET

General considerations

- RESET backs out uncommitted updates to ULTRA files opened by ULTRA statements, RDM user views opened by VIEW statements, OpenVMS RMS RU journal led external files opened by ACCESS statements and SQL databases connected to through EXEC_SQL statements.
- If the RELCHAIN MANTIS Option is set, then a RELEASE is performed each time the CHAIN statement without LEVELS is executed.
- If the MANTIS NORELCHAIN Option is set, then MANTIS is prevented from performing a RELEASE each time the CHAIN statement without LEVELS is executed.
- The RELCHAIN MANTIS Option must be set for IBM compatibility.
- **OpenVMS** RU Journal led file updates are backed out and delayed RELEASEs are performed.
- **OpenVMS** If the MANTIS File is RU journal led, RESET backs out uncommitted updates to MANTIS internal files.
- **OpenVMS** MANTIS automatically issues a START Recovery Unit system service when RMS Journaling is active for MANTIS and external files, and when an I/O request is issued and a recovery unit is not active.
- **OpenVMS** If you are using RMS Journaling, RESET causes ABORT Recovery Unit Processing.

- ♦ **OpenVMS** Your MANTIS program should ensure that a recovery unit is not active (before closing an external file) by issuing COMMIT or RESET prior to executing RELEASE.
- ♦ **OpenVMS** An RU journal led file cannot be closed when there are any uncommitted updates. Therefore, MANTIS external subroutines which execute ACCESS statements should perform COMMIT or RESET prior to returning via EXIT or RETURN; otherwise MANTIS may delay closing the file until the next implicit or explicit COMMIT or RESET.
- ♦ See also “COMMIT” on page 156.

Example

```
100 ULTRA REC("EXAMPLES:CUSTOMERS","CASINO")
  .
  .
  .
160 TRAP REC ON
170 INSERT REC
  .
  .
  .
220 UPDATE REC
230 IF REC="ERROR"
240 .RESET
250 END
  .
  .
  .
```

RETURN

Use the RETURN statement to return control from a subroutine or stop execution of a program.

RETURN

General considerations

- Use the RETURN statement when you want a subroutine to return control from any line within the ENTRY-EXIT statements. MANTIS returns control immediately as if the EXIT statement had been executed.
- If you use a RETURN statement in your main program, the effect is the same as a STOP statement. MANTIS returns to programming mode if the program was executing while in programming mode; or, MANTIS returns to your Facility Selection menu if the program was executing while not in programming mode.
- RETURN in an internal subroutine will return to the statement after the DO that invoked it.
- See also “STOP” on page 349, and “DO” on page 199.

Example

```
 10 ENTRY BROWSE
 20 .SCREEN MAP1("INDEX")
 30 .FILE REC1("INDEX","SERENDIPITY")
 40 .GET REC1
 50 .WHILE REC1="NEXT"
 60 ..CONVERSE MAP1
 70 ..IF MAP1="CANCEL"
 80 ...RETURN
 90 ..END
100 ..GET REC1
110 .END
120 EXIT
```

RND

Use the RND function to return a random real number in the range zero to “*a*,” excluding zero and “*a*.”

RND(*a*)

a

Description *Required*. Specifies the arithmetic expression you want as the lowest value outside of the range that can be returned as a random real number.

Format Any valid arithmetic expression

Consideration The expression result is coerced to BIG. Decimal expression results may therefore lose some precision or may generate an arithmetic fault (310) if the magnitude is too large for a BIG data type.

General consideration

- See also “SEED” on page 327.

Example

```
 10 ENTRY BUZZ_PHRASE_GENERATOR
 20 .DO SET_UP_VOCABULARY
 30 .HEAD "BUZZ PHRASE GENERATOR"
 40 .CLEAR
 50 .SHOW "I WILL GENERATE BUZZ PHRASES EACH TIME"
 60 .SHOW "YOU HIT 'RETURN'. ENTER 'END' TO STOP"
 70 .OBTAIN ANSWER
 80 .UNTIL ANSWER="END"
 90 ..X=1
100 ..WHILE X<22
110 ...A=INT(RND(10)+1)
120 ...B=INT(RND(10)+1)
130 ...C=INT(RND(10)+1)
140 ...SHOW FIRST(A) + "" + SECOND(B) + "" + THIRD(C)
150 ...X=X+1
  .
  .
  .
```

SCREEN

Use the SCREEN statement to define the screen you use in your program.

SCREEN *screen-name*(*libname*[,PREFIX]),...

screen-name

Description	*Required*. Specifies the symbolic name of the screen in subsequent CONVERSE statements.
Format	MANTIS symbolic name
Consideration	No processing occurs if *screen-name* is already defined.

libname

Description *Required*. Specifies the name of the screen design as it was saved during Screen Design.

Format Text expression that evaluates to a library name in the format:

```
[user-name:]screen-name
```

Consideration If the screen design is in your own library, you do not need to specify *user-name*.

PREFIX

Description *Optional*. Indicates that MANTIS place the prefix "*screen-name*_" on all variable names associated with this screen. For example, if the MANTIS screen BOTTLES has a field named VOLUME, the following statement causes the corresponding variable BIN_VOLUME to be defined in the work area:

```
10 SCREEN BIN("BOTTLES",PREFIX)
```

General consideration

- If you are using automatic mapping between screens and files or views, you may want to code your SCREEN statements after your ACCESS, VIEW, ULTRA, or INTERFACE statements. This ensures you are using the most current defined value for the field if you are a database driven shop. If one field is defined in two different ways, MANTIS uses the first definition it encounters in your program. Also, MANTIS automatically defines numeric fields in Screen Designs as BIG.

Example

```
 20 .FILE RECORD("INDEX","SERENDIPITY",16)
 30 .SCREEN_MAP("INDEX")
 40 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
 50 ..CLEAR MAP:LEVEL_NUMBER=1
 70 ..GET RECORD("WILLIAMS") LEVEL=LEVEL_NUMBER
 80 ..WHILE RECORD<>"END" AND LEVEL_NUMBER<16
 90 ...LEVEL_NUMBER=LEVEL_NUMBER+1
100 ...GET RECORD LEVEL=LEVEL_NUMBER
110 ..END
120 ..CONVERSE MAP
130 .END
```

screen-name

Use the screen-name function to return a text value identifying the last program function performed in response to the most recent CONVERSE statement using the screen name as the active map.

screen-name

screen-name

Description *Required*. Specifies the symbolic name of a screen as it is defined in a previously executed SCREEN statement.

General consideration

- ♦ The screen-name function returns the text value “TIMOUT” if the CONVERSE statement was specified with the TIME parameter and the CONVERSE terminated due to timeout (including the case TIME<=0).

Example

```
SCREEN MAP("INDEX")
CONVERSE MAP
SHOW MAP
```

SCROLL

Use the SCROLL statement to specify the display mode of the terminal or the amount by which windowing keys move the window.

$$\textbf{SCROLL}\begin{Bmatrix}\textbf{ON}\\ \textbf{OFF}\\ \textit{row}1\left[,\textit{col}1\right]\end{Bmatrix}$$

ON

OFF

Description *Optional*. Provided for compatibility with MANTIS for the IBM mainframe which allows alternate scrolling modes.

Consideration The SCROLL statement with ON or OFF has no effect and is included only for compatibility with MANTIS for the IBM mainframe which has alternate scrolling modes. MANTIS always scrolls the screen when in scroll mode. Each new line from a SHOW statement is displayed at the bottom of the screen after scrolling up the screen contents by one line. This can be used to minimize the amount of data transmitted to the terminal.

row1

Description *Optional*. Specifies the row increment for window movement. This is the number of rows that the window moves up or down in the logical display when you press function keypad keys 8 or 2 ({WINUP} or {WINDOWN}).

Format Arithmetic expression that evaluates to a value in the range of 0 through 255. MANTIS uses only the integer part of *row1*.

col1

Description *Optional*. Specifies the column increment for window movement. This is the number of columns that the window moves left or right in the logical display when you press function keypad keys 4 or 6 ({WINLEFT} or {WINRIGHT}).

Format Arithmetic expression that evaluates a value in the range of 0 through 255 Mantis uses only the integer part of *col1.*

General consideration

- ♦ Use the SCROLL statement with row1 and col1 parameters if you want to specify the window movement increments from your program. The increments can also be entered from the keyboard (preceded by i) in the Row and Column Coordinates Field of the window position map.

Example

```
10 SCREEN S("BIG_SCREEN")
20 SCROLL 11,40
30 CONVERSE S
```

SEED

Use the SEED statement to seed the random number generator (the RND function) to generate a new sequence of random numbers.

SEED

General considerations

- Without the SEED statement, the random number generator produces the same sequence of numbers each time you execute a program.
- With the SEED statement, the internal system clock seeds the random number generator which then produces an unpredictable sequence of numbers.
- See also “RND” on page 321.

Example

```
10 SEED
20 A = 10
30 B = RND(A)
```

SET

Use the SET statement to set a fault trap, to assign a value to a DCL symbol, to set a logical name, or to set a MANTIS option. Setting a fault trap causes a specified subroutine to be executed if a program fault occurs.

$$\textbf{SET}\left\{\begin{array}{l} \textbf{TRAP}\left\{\begin{array}{l} entry-name \\ program-name \end{array}\right\} \\ \\ \textbf{\$SYMBOL}\ name\ \textbf{TO}\ value \\ \textbf{\$LOGICAL}\ logical-name\ \left[,table\right]\textbf{TO}\ value \\ \textbf{\$OPTION}\ option-stmt \\ \\ \end{array}\right\}$$

TRAP

Description

Optional. Specifies a MANTIS subroutine to take control when a MANTIS fault occurs. Once a MANTIS fault handler is invoked, it must either handle the error (possibly ignoring it), or stop the program.

You can declare a fault handler at any MANTIS DOLEVEL. When a MANTIS fault occurs, the most recently declared handler is invoked (the handler at the highest DOLEVEL). When the handler EXITs or RETURNs, control resumes after the MANTIS program line in error, and always in the program context in which the fault handler was declared. This means that if a fault handler is declared in a MANTIS program, and an error occurs in an external subroutine, program control resumes in the calling program after the DO statement that invoked the faulty subroutine.

Format A MANTIS fault handler takes optional arguments as follows:

ENTRY *handler - name* (*p1* [, *p2* [, *p3* [, *p4*]]])

p1 Text parameter containing the MANTIS fault message. The first three characters reveal the MANTIS fault code (documented in *AD/Advantage MANTIS Messages and Codes OpenVMS/UNIX*, P39-1330).

p2 Text parameter containing the MANTIS line in error. For a bound program, the text of the program line is unavailable, and this argument is set to “At line nnnnn.” For an unbound program, the MANTIS line number and text are passed.

p3 Text parameter containing the last system error message saved by MANTIS. This parameter shows what would be returned by the HELP LAST command.

p4 Text parameter containing the name of the program in error. This is useful when the fault handler is at a lower DOLEVEL than the program causing the error. The name passed in this parameter is the same as that specified in the PROGRAM or CHAIN statement, or the RUN command, whichever was used to access the program originally.

Consideration **OpenVMS** The subroutine is called a fault handler, and unlike a OpenVMS condition handler, a MANTIS fault handler cannot “resignal” a condition to some other higher-level handler.

entry-name

Description *Optional*. Specifies the name of the internal subroutine to be executed if a program fault occurs.

Format MANTIS symbolic name defined in an ENTRY statement

program-name

Description *Optional*. Specifies the name of the external subroutine to be executed if a program fault occurs.

Format MANTIS symbolic name defined in a previously executed PROGRAM statement

$SYMBOL

Description *Optional*. Specifies the setting of a symbol. This symbol is available for the life of the process in which the MANTIS process is participating.

Specifies the value for a symbol with either a text or numeric expression, (but the attribute of the symbol created is text). However, symbols containing numeric strings can still participate in numeric operations.

Considerations

- **OpenVMS** MANTIS sets a OpenVMS local symbol.
- **UNIX** The symbol is a UNIX environment variable, $SYMBOL and $LOGICAL have the same effect in MANTIS for UNIX.

name

Description *Optional*. Specifies the name of a symbol to which you want to assign a value.

Format Text expression that evaluates to the name of a symbol

value

Description *Optional*. Specifies the value to which you want to assign to a symbol.

Format Text or arithmetic expression

$LOGICAL

Description *Optional*. Specifies the setting of a logical name. The logical name is only available to the MANTIS process, and is deleted when the MANTIS process is terminated.

logical-name

Description	*Optional*. Specifies a logical name.
Format	Text expression

Considerations

- Logical names should be specified in uppercase, as they are defined case-sensitive.
- **OpenVMS** DCL generally converts all command parameters to uppercase, so MANTIS likewise converts the logical name to uppercase before defining it.
- **OpenVMS** The logical name is defined in USER mode and in the PROCESS logical name table by default.

table

Description	*Optional*. Specifies a logical name table.
Format	A text expression
Default	LNM$PROCESS
Consideration	**UNIX** The *Optional* table name is not supported on UNIX.

value

Description	*Optional*. Specifies a value to which you want to assign the logical name.
Format	Text or arithmetic expression

$OPTION

Description	*Optional*. Specifies the setting of one or more MANTIS options.

option-stmt

Description	*Optional*. Specifies a MANTIS option source string.
Format	Text expression that evaluates to a MANTIS option source string
Consideration	The specified MANTIS option settings pertain to the MANTIS process only, and remain in effect (unless altered) until the MANTIS process is terminated.

General consideration

- ♦ See also “$OPTION” on page 292 and “$SYMBOL” on page 351.

Examples

```
10 SET TRAP FAULT
20 ACCESS F("FILE1","READ")
30 GET F (99)
40 STOP
50 ENTRY FAULT(ARG1,ARG2,ARG3)
60 SHOW "ARG1:";ARG1
70 SHOW "ARG2:";ARG2
80 SHOW "ARG3:";ARG3
90 EXIT
RUN
ARG1: 200 Record retrieval failed.
ARG2: 30 GET F KEY=99
ARG3: <system message that is associated with the fault>
SET $SYMBOL "MONTH" TO "NOVEMBER"
SHOW $SYMBOL("MONTH")
NOVEMBER
SET $OPTION "CRPOS/MAXDIMSIZE=1024/STRCOMPARE=ALPHANUMERIC"
SHOW $OPTION("CRPOS")
1
SHOW $OPTION("MAXDIMSIZE")
1024
SHOW $OPTION("STRCOMPARE")
ALPHANUMERIC
```

SGN

Use the SGN (“sign”) function to return -1 if “*a*” is less than 0, 0 if “*a*” equals 0, and +1 if “*a*” is greater than 0.

SGN(*a*)

a

Description *Required*. Specifies a numeric expression whose sign you want to determine.

Format Specifies any valid numeric expression

General consideration

- See also “NOT” on page 288, and “INT” on page 263.

Examples

```
SGN(-14)  returns -1
SGN(.00001)   returns +1
SGN(0)    returns 0
SGN(14E-17)   returns +1
```

SHOW

Use the SHOW statement to display data on the screen in scroll mode. You can specify data items which are to be displayed line by line with certain column position options. You can also use SHOW to display the fault trap subroutine.

$$\textbf{SHOW} \begin{bmatrix} \textit{data – item} \\ \textbf{,} \\ \textbf{;} \\ \textbf{AT } \textit{column} \\ \textbf{TRAP} \\ \textbf{RELEASE} \end{bmatrix} \ldots$$

data-item

Description *Optional*. Specifies a data item you want to display.

Format Arithmetic or text expression

Consideration MANTIS inserts the data item at the current column position and advances the current column position past the data item.

, (comma)

Description *Optional*. Advances the current column position to the next screen zone. Screen zones are 13 characters wide and start in columns 1, 14, 27, and so on.

; (semicolon)

Description *Optional*. Advances the current column position by one space.

AT *column*

Description *Optional*. Advances the current column position to the specified column number.

Format Arithmetic expression that evaluates to a valid column number

Consideration If the specified column number was less than the current column position, MANTIS terminates the current line and starts a new line.

TRAP

Description *Optional*. Displays the name of the fault trap subroutine, if any, for the current program or external subroutine.

RELEASE

Description *Optional*. Displays the full release of the MANTIS product.

Consideration This parameter should be used in preparation for a support call to Cincom Systems. The format of the output of this parameter is V2801.999.999.999. This release stamp will tell Cincom support in detail about all changes that went into this release.

General considerations

- ♦ A shorthand form of the SHOW statement is available. Entering a colon (:) at the start of the line is equivalent to SHOW. MANTIS changes the colon (:) to the word SHOW when you enter the program line.
- ♦ For scroll-mode output, MANTIS keeps track of the current column position which is where the first character of the next data item is inserted. MANTIS updates the current column position whenever a data item is inserted or a column position option is specified.
- ♦ When a SHOW statement ends with an item other than a comma or semicolon, MANTIS terminates and displays the current line, then starts a new line. When a SHOW statement ends with a comma or semicolon (called an unterminated SHOW statement), the next SHOW statement continues at the current column position.
- ♦ MANTIS also terminates the current line if it fills up. In this case, the line is displayed immediately on the screen.
- ♦ When a program terminates or stops at a breakpoint, any unterminated SHOW statement output is displayed.
- ♦ When SHOW is used as a command, the end of the command always terminates the current line which is displayed immediately.
- ♦ A SHOW statement with no parameters displays a blank line.
- ♦ If the MORE MANTIS Option is set, MANTIS displays “MORE” in the Reply Field after displaying a screenful of lines, and waits for you to reply before displaying any more lines. This prevents lines from being scrolled off the screen before you have had time to read them.

- If the NOMORE MANTIS Option is set, MANTIS scrolls without stopping.
- To terminate a multipage scroll-mode output display, you can enter KILL in the Reply Field. KILL terminates the executing program. Under program control, you can terminate multipage SHOWs by testing the KEY function after every SHOW statement.
- If no column position option (comma, semicolon, or AT) is specified between successive data items in a SHOW statement, MANTIS displays the data items with no intervening spaces.
- When an OBTAIN statement is executed, any unterminated SHOW statement output is displayed, and MANTIS positions the cursor after the current column position. This enables you to use a SHOW statement to display a prompt message at the bottom of the screen.
- When a CONVERSE statement is executed, any previously displayed SHOW statement output is overwritten by the full-screen mode display. If there is unterminated SHOW statement output, it is displayed in either a visible MESSAGE field or (if no such field exists) in the Message Field of the converse input map.
- When a PROMPT statement is executed, any previously displayed SHOW statement output is scrolled off the screen by the prompter. If there is unterminated SHOW statement output, it is displayed before the first line of the prompter.
- PROMPT and CONVERSE completely overwrite the screen, as does STOP if it returns to a facility program. You may need to precede these statements with a WAIT or OBTAIN statement to avoid overwriting lines displayed by previous SHOW statements before you have had time to read them. If your program returns to scroll mode by executing a SHOW statement after CONVERSE or PROMPT, the lines displayed by previous SHOW statements are again visible on the screen or at the bottom of the screen (in the scroll map).
- MANTIS displays numbers in standard decimal notation or in scientific notation, depending on their size (see “BIG and SMALL floating point data types” on page 53). Whichever notation is used, numbers never exceed 13 characters, the width of a screen zone. Use the FORMAT function to display numbers without forced rounding to the 13-character zone.

- See also “CONVERSE” on page 159,, “HEAD” on page 247,, “OBTAIN” on page 290, “PROMPT” on page 311, “SCROLL” on page 325, and “WAIT” on page 398.

Examples

```
 20 GET REC1
 30 WHILE REC1<>"END"
 40 .SHOW AT 20 NUMBER,FIRST_NAME;LAST_NAME
 50 .GET REC1
 60 END
100 SHOW "ENTER YOUR AGE";
110 OBTAIN AGE
120 SHOW AGE
RUN
ENTER YOUR AGE: 36   input is obtained from here
36
```

SIN

Use the SIN function to return the sine of angle “*a*” where “*a*” is in radians.

SIN(*a*)

a

Description	*Required*. Specifies the angle whose sine you want returned.
Format	Any valid arithmetic expression
Consideration	The expression result is coerced to BIG. Decimal expression results may therefore lose some precision or may generate an arithmetic fault (310) if the magnitude is too large for a BIG data type. If the angle is in degrees, it must be converted to radians. See “PI” on page 303 for how to do this.

General consideration

- See also “ATN” on page 110, “COS” on page 167, “PI” on page 303, and “TAN” on page 352.

Example

```
SHOW SIN(100)
-0.5063656411
```

SIZE

Use the SIZE function to return the size and dimensions of an expression, variable, or array.

$$\textbf{SIZE}\left(\left\{\begin{array}{l} \textit{string} \\ \left\{\begin{array}{l} \textit{text}-\textit{variable} \\ \textit{text}-\textit{array} \end{array}\right\}\textbf{,"MAX"} \\ \textit{variable}\textbf{,"DIM"} \\ \textit{array, dimension} \end{array}\right\}\right)$$

string

Description *Optional*. Tells MANTIS to return the current length of the expression you specify.

Format Text expression

Example

```
SHOW SIZE("NAME")    Returns 4

TEXT SURNAME(20)
STRING=”LEE”
SIZE(SURNAME) returns 3
```

text-variable,"MAX"
text-array,"MAX"

Description *Optional*. Tells MANTIS to return the maximum length (in characters) of the text variable or array whose name you specify.

Format Text variable or text array name

Consideration This parameter is invalid for numeric fields.

Example

```
TEXT SURNAME(20)     text variable 20 characters long:
TEXT NAME (3,25)     one-dimensional array with 3 occurrences of
  25 characters each.
SHOW SIZE(SURNAME, "MAX")       returns 20
SHOW SIZE(NAME, "MAX")   returns 25
SHOW SIZE(NAME(1), "MAX) returns 25
```

***variable*,"DIM"**

Description *Optional*. Tells MANTIS to return the number of array dimensions for a text or arithmetic variable.

Format MANTIS symbolic name

Consideration The value returned is zero (0) unless the specified variable is a text or numeric array.

Example

```
TEXT NAME(3,25)
BIG MATRIX(10,20)
SHOW SIZE(MATRIX,1)                         returns 10
SHOW SIZE(MATRIX,2)                         returns 20
SHOW SIZE(NAME,1)                                   returns 3
```

array, dimension

Description *Optional*. Tells MANTIS to return the number of occurrences in the specified array.

Format Array name and dimension number

Example

```
TEXT SURNAME(20),NAME(3,25)
BIG SCALAR,MATRIX(10,20)
SHOW SIZE (SURNAME,"DIM")                   returns 0
SHOW SIZE (NAME,"DIM")                      returns 1
SHOW SIZE (SCALAR,"DIM")          returns 0
SHOW SIZE (MATRIX,"DIM")          returns 2
```

Example

In the following example, SIZE is used to determine the total number of elements in an array. The routine sums the values in a one-dimensional numeric array and is called by an internal DO. The name of the array whose values are to be summed is passed to ARRAY_SUM, along with a previously defined BIG variable that holds the sum of the elements in the array. INDEX is used to select array elements starting with the highest subscript. Note that if the passed variable is not an array, zero is returned in SUM, and that INDEX is used as a conditional value (zero is FALSE, non-zero is TRUE)

```
DO ARRAY_SUM(name-of-array,sum-of-array)
 .
 .
 .
ENTRY ARRAY_SUM(ARRAY_NAME,SUM)
.SMALL INDEX
.INDEX=SIZE(ARRAY_NAME,1)
.SUM=0
.WHILE INDEX
..SUM=SUM+ARRAY_NAME(INDEX)
..INDEX=INDEX-1
.END
EXIT
```

SLICE

Use the SLICE statement to specify the number of statements in a program slice. With the SLOT value, this determines how many statements an application program can execute before being interrupted by MANTIS.

SLICE *statements*

statements

Description Specifies the number of statements in a program slice.

Format Arithmetic expression that evaluates to a value in the range 1–32767

General considerations

- ♦ The SLICE/SLOT statements are not in effect if running a BOUND version of the program. MANTIS maintains a statement counter in conjunction with the SLICE and SLOT statements but the counter is not incremented during execution of the bound version.
- ♦ See “SLOT” on page 343 and the SLOT MANTIS Option for more information about the effect of SLICE and SLOT on a running program.
- ♦ In MANTIS for the IBM mainframe, SLICE is effective in its own right, specifying the number of statements executed before CPU control is relinquished in favor of other system processes.
- ♦ This statement overrides the value of SLICE specified in your user profile and it is only reset when you sign on again or execute another SLICE statement.

Examples

```
10 SLICE 3000: SLOT 1
10 SLICE 300: SLOT 10
```

Either example allows execution of 3000 statements before MANTIS interrupts the program and issues a warning message. The examples assume that the SLOT MANTIS Option is set.

SLOT

Use the SLOT statement to specify the SLOT quota, which is the number of program slices an application program can execute before being interrupted by MANTIS.

SLOT *slices*

slices

Description Specifies the SLOT quota

Format Arithmetic expression which evaluates to a value in the range 1–32767

General considerations

- ♦ The SLICE/SLOT statements are not in effect if running a BOUND version of the program. MANTIS maintains a statement counter in conjunction with the SLICE and SLOT statements but the counter is not incremented during execution of the bound version.
- ♦ See the SLOT MANTIS Option for more information about the effect of SLICE and SLOT on a running program
- ♦ SLOT quota depletion causes premature program interruption depending on the circumstances. The SLOT MANTIS Option is disabled by default, but is automatically enabled when the IBM MANTIS Option is enabled.
- ♦ This statement overrides the value of SLOT specified in your user profile and it is only reset when you sign on again or execute another SLOT statement.
- ♦ If the CTRLC MANTIS Option is enabled you can stop a running program by pressing CTRL-C. In programming mode you will be able to resume program execution using either the GO or RUN command. This method of program control is not compatible with MANTIS for the IBM mainframe.

Examples

```
10 SLICE 3000: SLOT 1
10 SLICE 300: SLOT 10
```

Either example allows execution of 3000 statements before MANTIS interrupts the program and issues a warning message. The examples assume that the SLOT MANTIS Option is set.

SMALL

Use the SMALL statement to define numeric variables or arrays of numeric variables, allocating space in the work area to hold their values with up to seven significant digits. Values are initially set to zero.

SMALL *small-name*[(*dimension*,...)],...

small-name

Description	*Required*. Defines the symbolic name of the numeric variable or array.
Format	MANTIS symbolic name
Consideration	No processing occurs if *small-name* is already defined.

dimension

Description	*Optional*. Specifies an array dimension. Use one dimension parameter to specify a one-dimensional array, two dimension parameters to specify a two-dimensional array, and so on.
Format	Arithmetic expression that evaluates to a value in the range 1–*max*, where *max* is the value of the MAXDIMSIZE MANTIS Option.
Consideration	The maximum number of dimensions you can specify is determined by the value of the MAXNODIMS MANTIS Option. Each dimension specifies the maximum value of the corresponding array subscript.

General considerations

- ♦ The maximum number of variable names you can specify is determined by the value of the MAXVARS MANTIS Option.
- ♦ Binary operations involving mixed operand data types cause one operand to be converted to a compatible data type. The coercion rules are discussed in “Numeric considerations” on page 51.
- ♦ See also “BIG” on page 144, “DECIMAL” on page 175, and “INTEGER” on page 264.

- Auto mapping of variables takes place during complex variable declarations such as FILE and ACCESS statements. MANTIS generates a dissimilarity fault (212) if you attempt to map a new variable to an existing one with less precision. For example, if an ACCESS statement would define a DECIMAL variable with a precision of 10, and that variable is already defined in the work area with a precision of less than 10, a 212 fault will occur. For the purposes of dissimilarity fault checking, MANTIS data types are attributed with the following precision.

Type	Precision
SMALL	6
INTEGER	9
BIG	14
DECIMAL	1–31 as defined

So for example, it is illegal to auto-map an INTEGER or BIG variable onto an existing SMALL variable, and a SMALL variable cannot be mapped onto an existing DECIMAL(5) variable.

Example

```
10 X=15
20 SMALL ALPHA(64,3), BETA(X)
```

SQR

Use the SQR function to return the square root of an arithmetic expression.

SQR(*a*)

a

Description	*Required*. Specifies the arithmetic expression whose square root you want returned.
Format	Arithmetic expression
Consideration	The expression result is coerced to BIG. Decimal expression results may therefore lose some precision or may generate an arithmetic fault (310) if the magnitude is too large for a BIG data type.

General consideration

- ♦ See also **“**exponentiation (**)” on page 177, and “LOG” on page 278.

Example

```
SHOW SQR(100)
10
```

STATS

Use the STATS statement to return system process statistics in a numeric array.

STATS *array*

array

Description *Required*. Specifies a name for the numeric array.

Format MANTIS symbolic name, an INTEGER, SMALL, BIG or DECIMAL array with at least 9 elements

Considerations

- ♦ **OpenVMS** The numeric array must have at least nine elements for the following statistics, in this order:

Element	Name	Information
1	CPUTIM	CPU time in hundredths of a second
2	BUFIO	Buffered I/O count
3	DIRIO	Direct I/O count
4	PAGFL	Number of page faults
5	WSSIZE	Working set size
6	PPGCNT	Process page count—physical memory
7	GPGCNT	Global page count—physical memory
8	VIRTPEAK	Peak virtual page count—virtual memory
9	ELAPS	Elapsed time in seconds

For more information on elements 1–8, refer to the *VMS System Services Reference.* Refer to the $GETJPI system service.

- ♦ **OpenVMS** Each is a total since the OpenVMS process was activated.

- UNIX The numeric array must have at least nine elements for the following statistics, in this order:

Element	Name	Information
1	UTIME	User time
2	STIME	System time
3	CUTIME	Children User Time
4	CSTIME	Children System Time
5	UNUSED	
6	UNUSED	
7	UNUSED	
8	UNUSED	
9	ELAPS	Elapsed time in seconds

Example

```
 10 BIG STAT_START(9),STAT_END(9)
 20 STATS STAT_START
  .
  .
  .
100 STATS STAT_END
110 SHOW "ELAPSED SECONDS:";STAT_END(9)-STAT_START(9)
120 SHOW "CPU SECONDS:    ";(STAT_END(1)-STAT_START(1))/100
```

STOP

Use the STOP statement to terminate program execution and return you to programming mode if the program was executing while in programming mode; or, to your Facility Selection menu if the program was executing while not in programming mode.

```
STOP
```

General considerations

- ♦ In programming mode, a STOP in an external routine stops execution within the subroutine. Use the UP command to return to the calling program.
- ♦ It is not necessary to put a STOP statement at the end of the main program. MANTIS terminates execution of the program when it reaches an EXIT statement or when there are no more statements.
- ♦ See "Programming techniques" on page 415 for additional information on the STOP statement and how it works with external DO.
- ♦ See also "CHAIN" on page 150, EXIT, "RETURN" on page 320, and "DO" on page 199.

Examples

```
 10 ENTRY MAIN
 20 .SET TRAP FAULT_HANDLER
 30 ...
 40 ...
 50 EXIT
 60 ENTRY FAULT_HANDLER
 70 ...
 80 ... determine cause of fault
 90 ...
100 IF CANT_RECOVER_FROM_FAULT
110 .SHOW ERRMESS:SHOW PROGLINE:SHOW RMSERR
120 .STOP
130 END
140 EXIT
```

```
 10 WHILE CONDITION1
 20 .WHILE CONDITION2
 30 ..WHILE CONDITION3
 40 ...WHILE CONDITION4
 50 ...
 60 ...
 70 ...
 80 ....IF WANT_TO_TERMINATE_IN_A_HURRY
 90 .....STOP
100 ....END
110 ...END
120 ..END
130 .END
140 END
```

$SYMBOL

Use the $SYMBOL function to return the text value assigned to the symbol name specified.

$SYMBOL(*name*)

name

Description *Required*. Specifies the symbol name whose text value you want returned.

Format Text expression

Consideration **UNIX** The symbol is a UNIX environment variable, $SYMBOL and $LOGICAL have the same effect in MANTIS for UNIX.

Example

```
SET $SYMBOL "MONTH" TO "NOVEMBER"
SHOW $SYMBOL("MONTH")
NOVEMBER
```

TAN

Use the TAN function to return the tangent of angle “*a*” where “*a*” is in radians.

TAN(*a*)

a

Description *Required*. Specifies the angle (in radians) whose tangent you want returned.

Format Arithmetic expression

General considerations

- ♦ The expression result is coerced to BIG. Decimal expression results may therefore lose some precision or may generate an arithmetic fault (310) if the magnitude is too large for a BIG data type.
- ♦ See also “ATN” on page 110, “COS” on page 167, “PI” on page 303, and “SIN” on page 338.

Example

```
100 Z=X**3+Y
.
.
.
200 CORNER=TAN(Z)
```

TERMINAL

Use the TERMINAL function to return a text value containing the terminal ID.

TERMINAL

General consideration

- ♦ See also “ATTRIBUTE (statement)” on page 111.

OpenVMS Example

```
SHOW TERMINAL
_VTA65:
```

UNIX Example

```
SHOW TERMINAL
/dev/tty01
```

TERMSIZE

Use the TERMSIZE function to return a text value giving the size of the terminal in rows and columns in the format RRXCCC.

TERMSIZE

General consideration

- ♦ See also “ATTRIBUTE (statement)” on page 111.

Examples

TERMSIZE returns (physical) rows and columns for the current terminal.

```
SHOW  TERMSIZE
24X80
```

TERMSIZE(1,2) returns row dimensions for the current terminal.

```
SHOW  TERMSIZE(1,2)
24
```

TERMSIZE(4) returns columns for the current terminal.

```
SHOW  TERMSIZE(4)
080
```

TEXT

Use the TEXT statement to define text variables or arrays of text variables. The current lengths are initially set to zero.

TEXT *text - name* [([*dimension*,**...**,] *length*)]**,...**

text-name

Description *Required*. Specifies the symbolic name of the text variable or array.

Format MANTIS symbolic name

Consideration No processing occurs if *text-name* is already defined.

dimension

Description *Optional*. Specifies an array dimension. Use one dimension parameter to specify a one-dimensional array, two dimension parameters for a two-dimensional array, and so on.

Format Arithmetic expression that evaluates to a value in the range 1–*max*, where *max* is the value of the MAXDIMSIZE MANTIS Option.

General considerations

- The maximum number of dimensions you can specify is determined by the value of the MAXNODIMS MANTIS Option. Each dimension specifies the maximum value of the corresponding array subscript.
- If MANTIS encounters a previously defined name, it is not processed again.

length

Description *Optional*. Specifies the maximum length of a text variable or array element.

Default 16 characters

Format Arithmetic expression that evaluates to a value in the range 1–*max*, where *max* is the value of the MAXSTRLEN MANTIS Option.

General consideration

- ♦ The maximum number of variable names you can specify is determined by the value of the MAXVARS MANTIS Option.

Examples

```
20 TEXT ALPHA(64,3), BETA(12)
```

Defines a text array ALPHA with 64 3-character elements and a 12-character text variable BETA.

MANTIS accepts only as many characters in a text variable as you specify in the TEXT statement. For example, if you enter:

```
10 TEXT GAMMA, DELTA(5)
20 GAMMA="123456789ABCDEFGHIJK": DELTA="123456789"
30 SHOW GAMMA, DELTA
```

the screen displays:

```
123456789ABCDEFG 12345
```

TIME (function)

Use the TIME function to return a text string containing the current time. The format of the time is determined by a previous time mask specification using the TIME statement

TIME

General considerations

- The current time mask setting is inherited on external DO and CHAIN LEVEL, and is restored upon subprogram EXIT. This inheritance does not occur when a non-privileged user's program attempts to externally DO or CHAIN LEVEL a privileged user's program.
- A CHAIN (without LEVEL) will reset the time mask to NULL (“”).
- For non-privileged users' programs, if the time mask has not been previously specified, or has been set to NULL (“”), the TIME function returns the current time in the format specified by the TIMEMASK MANTIS Option. If the TIMEMASK MANTIS Option value is also NULL (“”) then HH:MM:SS format is used.
- For privileged users' programs, if the time mask has not been previously specified, or has been set to NULL (“”), the TIME function returns the current time in HH:MM:SS format.
- See also “DATE (function)” on page 171, “DATE (statement)” on page 173, and “TIME (statement)” on page 357.

Examples

```
TIME="HH:MM"
SHOW TIME
11:25
  .
30 ..TIME="HH:MM:SS"
40 ..CUTOFF="10:45:00"
50 ..IF TIME>CUTOFF
60 ...SHIP_DATE=TOMORROW
70 ..ELSE
80 ...SHIP_DATE=DATE
90 ..END
  .
```

TIME (statement)

Use the TIME statement to specify the time mask to be used by the TIME function.

TIME=*mask-expression*

mask-expression

Description	*Required*. Specifies the time mask to be used by the TIME function.
Format	A text expression of 0–255 characters

Considerations

- If the text string specified is longer than 255 characters, the first 255 characters will be used.
- The following special strings are substituted with the data described when the TIME function is invoked. Any characters in the mask that are not part of one of these special strings are treated as punctuation and are not translated:

  AM = AM/PM indicator

  HH = hours

    12 hour clock (zero suppressed) if AM is specified

    24 hour clock if AM not specified

  MM = minutes

  SS = seconds

- Where there are ambiguities on substitution, the substitution takes place in the order of the special strings listed above.
- There is a special mask escape character (%). Any character following this escape is taken literally and the escape can be used to break up what might otherwise be a valid substitution string.
- When alphabetic substitution is required (for example, AM), the case of any special string characters are reproduced on substitution. Case is not important for numeric substitution.

- No subscripting of the statement is permitted. (TIME(1,5)=“HH:MM” is invalid.). However, HOURS=TIME(1,2) is a valid use of the TIME function.

General consideration

- See also “TIME (function)” on page 356 “TIME (statement)” on page 357, “DATE (function)” on page 171, and “DATE (statement)” on page 173.

Examples

```
TIME="HH:MM:SS am"
SHOW TIME
 5:45:12 pm
TIME="24 hour time is HH:MM:SS"
24 hour time is 17:45:12
TIME="Sa%m said it was HH o'clock AM"
SHOW TIME(1,-4)
Sam said it was 5 o'clock
```

TRAP

Use the TRAP statement to determine whether error conditions in GET, UPDATE, INSERT, and DELETE processing terminate program execution or whether MANTIS provides the program with an error status and allows execution to continue.

$$\textbf{TRAP} \begin{Bmatrix} \textit{file} - \textit{name} \\ \textit{access} - \textit{name} \\ \textit{ultra} - \textit{name} \\ \textit{view} - \textit{name} \end{Bmatrix} \quad \begin{bmatrix} \underline{\textbf{ON}} \\ \textbf{OFF} \end{bmatrix}$$

file-name

Description	*Optional*. Specifies the symbolic name of the MANTIS file.
Format	MANTIS symbolic name as defined in a previously executed FILE statement

access-name

Description	*Optional*. Specifies the symbolic name of the external file.
Format	MANTIS symbolic name as defined in a previously executed ACCESS statement

ultra-name

Restriction	SUPRA PDM users only.
Description	*Optional*. Specifies the symbolic name of the ULTRA file.
Format	MANTIS symbolic name as defined in a previously executed ULTRA statement

view-name

Restriction	SUPRA RDM users only.
Description	*Optional*. Specifies the name of the RDM user view.
Format	MANTIS symbolic name as defined in a previously executed VIEW statement.

ON

OFF

Description *Optional*. Indicates whether program execution should continue after an error message with an error status (ON) or terminate with an error message (OFF).

Default ON

General considerations

- TRAP...ON enables MANTIS to return a status to the program after execution of a GET, UPDATE, INSERT, or DELETE statement. The range of statuses is listed below. These status values do not necessarily apply to all file types.

  DATA A conversion error occurred transforming a field between internal and external formats.

  DUPLICATE A record or row with the same key already exists.

  ERROR DELETE EXTERNAL FILES: A physical error occurred while deleting the record, or the RRN for relative files specified a record number that does not exist.

  - GET EXTERNAL FILES: A physical error occurred while retrieving the record. The RFA for a sequential file was not at a record boundary, or the RRN for a relative file specified a record number outside the file range.
  - INSERT EXTERNAL FILES: A physical error occurred while inserting the record, the file was full, or an indexed or relative file contained a record with the same key or RRN as the record to be inserted.
  - UPDATE EXTERNAL FILE: A physical error occurred while updating the record, or the record to be updated did not exist on the file.

  HELD MANTIS failed to perform the I/O request because a required record or row was held by another user.

  LOCK The password specified in the FILE, ACCESS, or ULTRA statement is not valid.

NOTFOUND   The chain set in a related file with the requested key does not exist.

SETS   You cannot delete a record in a primary file until you delete the associated chains of records in related files.

- See “DBCS support feature” on page 455 for more information on RDM error status functions.
- If you do not include a TRAP statement in your program, any error condition occurring during GET, UPDATE, INSERT, or DELETE processing terminates execution (as if TRAP OFF had been executed).
- This statement is required only in very special circumstances. The program assumes all responsibility for detecting and handling error situations when TRAP is used. You should be cautious about your use of TRAP. If you do not take the appropriate action for trapped errors, you may lose database integrity. Consult your Master User before you use the TRAP statement.

Example

```
100 FILE REC("EXAMPLES:CUSTOMERS","CASINO")
    .
    .
    .
160 TRAP REC ON
170 INSERT REC
    .
    .
    .
220 UPDATE REC
230 IF REC = "ERROR"
240 .MSG="UPDATE FAILED"
250 END
    .
    .
```

TRUE

Use the TRUE function to return the value TRUE. TRUE returns a numeric value of 1.

TRUE

General considerations

- ♦ Any nonzero value in a relational expression evaluates to "TRUE" condition. For example, you can code IF ERROR_CONDITION and not IF ERROR_CONDITION = TRUE. The latter will be false if ERROR_CONDITION has any value except +1.
- ♦ See also "FALSE" on page 209, , "NOT" on page 288, and , "ZERO" on page 402.

Example

```
10 ENTRY CHECK(FIELD,STATUS)
20 .STATUS=FALSE
30 .COUNT=SIZE(FIELD)
40 .WHEN FIELD=SPACES(1,COUNT)
50 ..STATUS=TRUE
60 .END
70 EXIT
```

TXT

Use the TXT function to return the text value of “*a*” in numeric display format.

TXT(*a*)

a

Description *Required*. Specifies the arithmetic expression whose text value you want returned.

Format Arithmetic expression

General considerations

- By default TXT(0) returns a single space; however, if you wish anything other than a space to be returned, you may specify it by using the TXT0 MANTIS Option . It does not return an empty string (NULL).
- MANTIS in a non-IBM environment does place a leading “0” before nonzero values between -1 and 1. IBM MANTIS does not place a leading “0” before nonzero values between -1 and 1.
- The TXT0 MANTIS Option must be set to space for IBM compatibility
- See also the “FORMAT” function on page 217. FORMAT allows masking to control the returned text.

Example

```
SHOW TXT(100)
100
```

ULTRA

NOTE

This statement applies to SUPRA PDM users only.

Use the ULTRA statement to sign-on or sign-off a SUPRA PDM database or to open an ULTRA file view to a SUPRA PDM File. MANTIS retrieves the ULTRA File View from the library and defines a MANTIS variable or array with the same name as each field in the view (if the ULTRA File View is consistent with the ULTRA Database Description).

$$\textbf{ULTRA}\ \left\{\begin{array}{l} \textit{ultra-name}\ (\textit{libname}, \textit{password}[,\textbf{PREFIX}]\ [,\textit{levels}]) \\ \textbf{ON}\ [(\textit{dbmod})] \\ \textbf{OFF} \end{array}\right\}$$

ultra-name

Description *Required*. Specifies the symbolic name of the SUPRA PDM files in subsequent GET, UPDATE, INSERT, DELETE, and DEQUEUE statements.

Format Unique MANTIS symbolic name

Consideration No processing takes place if *ultra-name* has already been defined.

libname

Description *Required*. Specifies the name of the PDM files profile as it was saved during ULTRA File View Design.

Format Text expression that evaluates to a library name in the format:

```
[user-name:]ultra-file-name
```

Consideration If the PDM files profile is in your own library, you do not need to specify *user-name*.

password

Description *Required*. Specifies the password needed for the type of access to the PDM files (GET, UPDATE, INSERT/DELETE) your program requires.

Format Text expression that evaluates to the password specified during ULTRA File View Design

PREFIX

Description *Optional*. Indicates that MANTIS place the prefix “*ultra-name*_” on all variable names associated with this PDM files. For example, if the PDM files BOTTLES has a field named VOLUME, the following statement causes the corresponding variable BIN_VOLUME to be defined in the work area:

```
10 ULTRA BIN(”BOTTLES”,”MAGIC”,PREFIX)
```

Consideration MANTIS also prefixes the reference variables specified in the ULTRA File View Design record layout.

levels

Description *Optional*. Specifies the number of levels you want to use in GET, UPDATE, INSERT, or DELETE statements for this PDM file.

Format Arithmetic expression that evaluates to a value in the range 1–255

Considerations

- If you specify *levels*, you must also specify LEVEL=*level-number* (unless you want the default LEVEL=1) in all GET, UPDATE, INSERT, and DELETE statements for the PDM files.
- If you do not specify *levels*, MANTIS defines an INTEGER, SMALL, BIG, DECIMAL or TEXT variable or array for each field in the ULTRA File View Design record layout, according to the type and dimensions specified during ULTRA File View Design.
- If you specify *levels*, MANTIS adds an extra dimension to each field to allow a separate value to be stored at each level. MANTIS defines a one-dimensional array instead of a variable, and a two-dimensional array instead of a one-dimensional array, and so on. The first dimension of each array is *levels*.

ON

Description *Optional*. Specifies a request to sign-on to SUPRA PDM using either a specified DBMOD or the DBMOD to which it was last connected.

Considerations

- If MANTIS is already connected to SUPRA PDM, the ULTRA ON statement signs off before attempting to sign-on.
- You can use ULTRA ON after a CALL to an INTERFACE program to sign-on again to the PDM (assuming you used ULTRA OFF before the CALL, or that the INTERFACE program performed a sign-off).
- The ULTRA ON statement is not compatible with MANTIS for the IBM mainframe.

dbmod

Description *Optional*. Specifies the DBMOD to sign-on to. See the General considerations below to see how the DBMOD identifier is processed.

Default The last DBMOD identifier used by MANTIS in signing on to SUPRA PDM. If one does not exist, a program fault occurs.

Format A TEXT expression giving a 6-character DBMOD identifier

Considerations

- A COMMIT is performed prior to the sign-off.
- The DBMOD specified in an ULTRA file view is a 6-character identifier, XXXXXX. If the logical name MANTIS_DBMOD_XXXXXX exists, it is translated to obtain the true DBMOD identifier, say, YYYYYY. In either case the DBMOD file must exist either in the current directory as XXXXXX.MOD or YYYYYY.MOD, or, XXXXXX or YYYYYY must be a logical name for the file specification of the DBMOD.

OFF

Description *Optional*. Specifies a request to sign-off from SUPRA PDM.

Considerations

- You can use ULTRA OFF prior to a CALL statement to enable an INTERFACE program to sign-on to the PDM.
- The ULTRA OFF statement is not compatible with MANTIS for the IBM mainframe.

General considerations

- MANTIS can only connect to one SUPRA database (DBMOD) at a time, so all ULTRA file views open at the same time must specify the same DBMOD.
- If a view statement is processed before an ULTRA statement, all PDM files in the database are opened in SHRE mode when the view statement is executed.
- A view statement cannot be processed after an ULTRA statement, since MANTIS cannot sign onto RDM when it has already signed onto PDM.
- While MANTIS is connected to the PDM, further ULTRA statements for the same database do not cause a sign-on to occur. If the current connection specifies RDLY access to some PDM files, and another ULTRA statement specifies UPDE access to a PDM file, the access may be denied by the PDM by virtue of the existing connection.
- You cannot sign on to multiple PDM databases concurrently. The ULTRA OFF/ON statements allow a single program to sign on to different databases consecutively.

- If you use ULTRA OFF/ON statements to allow your program to access more than one PDM database, RDLY access to PDM files may not be enforced at the PDM level. If you require the PDM to enforce RDLY access to any data sets, do not attempt to interleaf access to different PDM databases, for example:

  ```
  ULTRA CUST(“CUST”,”READ_ONLY”)
  ULTRA OFF
  ULTRA EMPL(“EMPL”,”READ_ONLY”)
  GET EMPL
  ULTRA ON(“CUSTDB”)
  GET CUST
  ```

  Ensure that all ULTRA statements and I/O statements for a particular database are completed. You can then use ULTRA OFF before ULTRA and I/O statements on the next database. ULTRA OFF/ON can be used with less discretion if RDLY access need not be enforced by the PDM. MANTIS enforces read-only access but only on the data sets specified in ULTRA file views, not any related data sets that the view may access.

- After an ULTRA OFF, the next GET, UPDATE, INSERT, or DELETE statement automatically signs-on to SUPRA PDM, if necessary, using the previous DBMOD.

- When the ULTRA statement is processed MANTIS validates the ULTRA file view. Data set details are read from the DBMOD file unless the view was created or modified after the last compilation of the DBMOD. The statement can be optimized by replacing existing ULTRA file views whenever the DBMOD is recompiled.

Example

```
40 ULTRA CUSTOMERS(“CLIENT”,”SALES”)
50 ULTRA HISTORY(“PAYMENTS”,”TEXAS”,11)
```

If the SALES password allows viewing only, MANTIS opens the CUSTOMERS file without ULTRA update access. HISTORY would be allocated 11 levels.

ultra-name

Use the ultra-name function to return a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on an ULTRA file with the symbolic name “ultra-name.”

ultra-name

ultra-name

Description *Required*. Specifies the symbolic name of the ULTRA file.

Format MANTIS symbolic name as defined in a previously executed ULTRA statement

Example

```
40 ULTRA CUSTOMERS(“CLIENT”,”SALES”)
.
.
.
SHOW CUSTOMERS
```

UNPAD

Use the UNPAD statement to remove padding characters from a text variable, array element, or substring.

$$\textbf{UNPAD}\ \textit{string}\ [\textit{padding}]\ \begin{bmatrix}\textbf{BEFORE}\\ \underline{\textbf{AFTER}}\\ \textbf{ALL}\end{bmatrix}$$

string

Description *Required*. Specifies the text variable, array element, or substring from which you want to remove padding characters.

Format Name of a TEXT variable, the subscripted name of a TEXT array element, or a reference to a TEXT substring

Considerations

- When the string is not a substring, the UNPAD statement removes all consecutive characters that match the padding character from the specified end(s) of the string. The current length of the string is reduced accordingly.
- If the referenced variable is subscripted (apart from the array occurrence) (see “LET” on page 273), the BEFORE, AFTER, and ALL options cannot be used. If you try to use the substring subscripts with one of these options, you receive an error message.
- If the referenced variable has two substring subscripts, each subscript represents the boundaries of the unpad operation. The leftmost boundary is marked by the first substring subscript; the rightmost boundary by the second substring subscript.
- If the referenced variable has one substring subscript, MANTIS assumes that the second (missing) subscript is equal to the current size of the variable. Therefore, the first substring subscript marks the starting point of the unpad operation. With no second substring subscript, the end of the unpad operation is the rightmost byte of the originally defined area for the variable.
- MANTIS unpads the variable by comparing each character in the variable with the unpad character. If the two are equal, the matching character is stripped off and the rest of the text is moved to the left. The current length is reduced by 1. This process repeats until a character not equal to the unpad character is detected. Characters are checked from left to right if you use the BEFORE option; from right to left if you use the AFTER option; and a combination with the ALL option.

padding

Description	*Optional*. Specifies the padding character.
Format	Text expression
Default	If you do not specify a padding character, MANTIS removes spaces.
Consideration	The first character is the padding character. Any subsequent characters are ignored.

BEFORE

Description	*Optional*. Removes padding characters from the left-hand end of the string.

AFTER

Description *Optional*. Removes padding characters from the right-hand end of the string.

ALL

Description *Optional*. Removes padding characters from both ends of the string.

Consideration If you do not specify BEFORE, AFTER, or ALL, MANTIS removes padding characters from the right-hand end of the string.

General considerations

- ♦ You cannot use BEFORE, AFTER, or ALL when removing padding from a substring.
- ♦ See also “PAD” on page 296.
- ♦ You can remove padding from a substring by specifying one or two subscripts (see “Text substrings” on page 45):
  - If the first and last character positions are specified, consecutive padding characters are removed from the right-hand end of the substring and any characters to the right of the substring are moved left to close the gap.
  - If only the first character position is specified, the last position defaults to the last character in the string and consecutive padding characters are removed from the right-hand side of the substring.

  In either case, the current length of the string is reduced accordingly.

Examples

Strip leading and/or trailing blanks. If *string* has the contents shown below, the result of unpadding is shown for each type of unpadding supported when no pad character is specified. The contents of *string* are changed by the UNPAD statement.

Contents	Unpadding specified	Result
“ JOE JACKSON ”	AFTER	“ JOE JACKSON”
“ JOE JACKSON ”	BEFORE	“JOE JACKSON ”
“ JOE JACKSON ”	ALL	“JOE JACKSON”

Determine if a text field is all blanks or null:

```
UNPAD CLIENT_NAME
IF CLIENT_NAME=NULL
.
.
.
END
```

Determine if a text field is all zeros or null:

```
UNPAD CLIENT_NUMBER “0”
IF CLIENT_NUMBER=NULL
.
.
.
END
```

UNTIL-END

Use the UNTIL-END statement to execute a block of statements repeatedly until a specified condition becomes true. MANTIS executes the block of statements in the range of the UNTIL-END once before it tests the condition.

UNTIL *expression*

.

statements

.

END

expression

Description *Required*. Specifies the condition which ends execution of the UNTIL loop when true.

Format Expression which evaluates to TRUE (non-zero) or FALSE (zero)

General considerations

- UNTIL-END executes the enclosed statements at least once. If you need to omit execution of *statements* if the initial condition is not met, use the WHILE-END statement.
- The UNTIL statement condition is not evaluated before the statements are executed.
- Each of the statements UNTIL and END must appear on a line by itself. Only a comment (separated by a colon) can follow the UNTIL expression.
- See also "FOR-END" on page 214and , "WHILE-END" on page 401.

Example

```
10 UNTIL YEAR = 2000
20 .SHOW "ENTER YEAR IN FORMAT YYYY"
30 .OBTAIN YEAR
40 END
```

UPDATE

Use the UPDATE statement to update the contents of a record in a MANTIS file, an external file, an ULTRA file, or a row in an RDM user view. The UPDATE statement may also be used to update all data fields of a screen that is currently in the mapset.

UPDATE (MANTIS file)

UPDATE *file-name* [LEVEL=*level-number*]

file-name

Description	*Required*. Specifies the symbolic name of the MANTIS file you want to update.
Format	MANTIS symbolic name as defined in a previously executed FILE statement

LEVEL=*level-number*

Description	*Optional*. Specifies which level of MANTIS array elements contains the updated values of the record contents.
Default	LEVEL=1
Format	Arithmetic expression which evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding FILE statement
Consideration	You must specify LEVEL=*level-number* if the corresponding FILE statement specifies *levels* unless you want the default value.

General considerations

- You do not need to read a record from a MANTIS file before updating it.
- MANTIS obtains the value of each field in the file from the corresponding MANTIS variable or array element and writes it to the record indicated by the key fields.

- ♦ You can obtain the status of the UPDATE by using the “file-name” built-in text function. The status is a null text value if the UPDATE was successful. If the UPDATE is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the system error message that may have been returned. This information is also available to MANTIS fault handling routines (see the SET TRAP statement). MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

  LOCK The password specified in the FILE statement for this file view is not valid for updates.

  ERROR A physical error occurred while updating the record; or the record to be updated did not exist on the file.

- ♦ Do not change a key field in a record before an UPDATE; instead, delete the existing record and insert the new record with an altered key.

- ♦ MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

- ♦ **OpenVMS** If RMS Journaling is active and the file is RU Journal, the following occurs:

  The START_RU (Recovery Unit) occurs.

  - The record is locked until COMMIT or RESET.
  - You may back out the update via RESET.
  - During the START_RU, if RMS Journaling is active, all RU Journal led files opened for update (including the MANTIS File) automatically join the Recovery Unit. This implies that a release of any of these files is delayed until the next COMMIT or RESET.

- ♦ **OpenVMS** If RMS Journaling is not active, and the file is RU Journal led, MANTIS issues a fault. UPDATE (EXTERNAL FILE)

Example

```
60 ..CONVERSE MAP
70 ..WHEN MAP="PF1"
80 …INSERT RECORD
90 ..WHEN MAP="PF2"
100 …DELETE RECORD
110 ..WHEN MAP="PF3"
120 …UPDATE RECORD
130 ..END
140 ..GET RECORD
150 .END
160  EXIT
```

UPDATE (external file)

UPDATE *access-name* [LEVEL=*level-number*]

access-name

Description *Required*. Specifies the symbolic name of the external file you want to update.

Format MANTIS symbolic name as defined in a previously executed ACCESS statement

LEVEL=*level-number*

Description *Optional*. Specifies which level of MANTIS array elements contains the updated values of the record contents.

Default LEVEL=1

Format Arithmetic expression which evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ACCESS statement

Consideration You must specify LEVEL=*level-number* if the corresponding ACCESS statement specifies *levels* unless you want the default value.

General considerations

- You do not need to read the record from an indexed file before you update it; but, before you issue the UPDATE statement, you should make sure that the MANTIS variables for each field in the external file view contain the right values. You must, however, read the record (using GET) for sequential and relative files before issuing the UPDATE statement.

- MANTIS obtains the value of each field in the external file view from the corresponding MANTIS variable or array element and writes it to the record to be updated. If an external file view does not define fields which exist in the external file, the values of those undefined fields will not be kept when updated.

- For indexed files, the contents of key data elements identify the record to be updated.

- Do not change a key field in a record in an indexed file; instead, delete the existing record and insert the new record with an altered key.

- For sequential files, the associated reference variable (which must be defined during External File View Design) contains the Record's File Address (RFA) which identifies the record to be updated. You cannot update a variable-length record if you have changed its length.

- For relative files, the associated reference variable (defined during External File View Design) contains the Relative Record Number (RRN) which identifies the record to be updated.

- You can obtain the status of the UPDATE by using the "access-name" built-in text function. The status is a null text value if the UPDATE was successful. If the UPDATE is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the system error message that may have been returned. This information is also available to MANTIS fault handling routines (see the SET TRAP statement). MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

  LOCK The password specified in the ACCESS statement for this file view is not valid for updates.

  ERROR A physical error occurred while updating the record or the record to be updated did not exist on the file.

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

- **OpenVMS** If RMS Journaling is active and the file is RU Journal led, the following occurs:

  The START_RU (Recovery Unit) occurs.

  - The record is locked until COMMIT or RESET.

  - You may back out the UPDATE via RESET.

  - During the START_RU, if RMS Journaling is active, all RU Journal led files opened for update (including the MANTIS File) automatically join the Recovery Unit. This implies that a release of any of these files is delayed until the next COMMIT or RESET.

- **OpenVMS** If RMS Journaling is not active, and the file is RU Journal led, MANTIS issues a fault.

Example

```
 20 .ACCESS RECORD(“INDEX”,”SERENDIPITY”,16)
 30 .SCREEN MAP(“INDEX”)
 40 .CONVERSE MAP
 50 .COUNTER=1
 60 .WHILE MAP<>”CANCEL” AND COUNTER<17
 70 ..WHEN INDICATOR(COUNTER)=”G”
 80 …GET RECORD LEVEL=COUNTER
 90 ..WHEN INDICATOR(COUNTER)=”U”
100 …UPDATE RECORD LEVEL=COUNTER
```

UPDATE (screen)

UPDATE *screen-name*

screen-name

Description	*Required*. Specifies the symbolic name of the screen you want to update.
Format	MANTIS symbolic name as defined in a previously executed SCREEN statement.

General considerations

- The specified screen must be in the mapset.
- MANTIS updates the screen's visible data fields according to the fields' attributes and values of the corresponding MANTIS variables.
- If the mapset has been altered since the last CONVERSE with I/O (for example, by adding/removing/moving screens), the UPDATE will result in a full refresh of the mapset.
- The UPDATE does not effect the "screen-name" built-in text function.
- Unlike the CONVERSE statement, the UPDATE statement may be used in Field Entry and Field Exit Routines.

Example

```
   .
 30 SCREEN S(“INSTALL”)
 40 SCREEN P(“PERCENT_COMPLETE”)
   .
   .
100 CONVERSE S
110 IF S<>”CANCEL”
120  PERCENT=0
130  PERCENT_INC=5
140  PERCENT_BAR_INC=PERCENT_INC/100*SIZE(PERCENT_BAR,”MAX”)
150  PERCENT_BAR_X=PERCENT_BAR_INC
160  CONVERSE P(10,10) SET TIME=0
170  UNTIL PERCENT=100
   .
   .
300   PERCENT=PERCENT+PERCENT_INC
310   PERCENT_BAR(1,PERCENT_BAR_X)=” “
320   PERCENT_BAR_X=PERCENT_BAR_X+PERCENT_BAR_INC
330   UPDATE P
340  END
350 ELSE
360  SHOW “INSTALLATION HAS BEEN TERMINATED”
370  STOP
380 END
   .
   .
```

UPDATE (ULTRA file view)

This statement applies to SUPRA PDM users only.

UPDATE *ultra-name* [LEVEL=*level-number*]

ultra-name

Description	*Required*. Specifies the symbolic name of the ULTRA file you want to update.
Format	MANTIS symbolic name as defined in a previously executed ULTRA statement

LEVEL=*level-number*

Description	*Optional*. Specifies which level of MANTIS array elements contains the updated values of the record contents and reference variable.
Default	LEVEL=1
Format	Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ULTRA statement
Consideration	You must specify LEVEL=*level-number* if the corresponding ULTRA statement specifies *levels* unless you want the default value.

General considerations

♦ You do not need to read the record before you update it. MANTIS always obtains exclusive control of the record before requesting the database to update it.

♦ MANTIS obtains the value of each field in the ULTRA file view from the corresponding MANTIS variable or array element and writes it to the record to be updated.

♦ Values must be assigned to all fields in the view; otherwise, MANTIS overlays existing data with blanks or zeros.

- If you change any keys in a related file record, MANTIS places the updated record in the chain set associated with the new key values. The password supplied on the ULTRA statement must authorize INSERT/DELETE to allow this to happen.

- You can obtain the status of the UPDATE by using the “ultra-name” built-in text function. The values are GOOD and HELD.

  GOOD MANTIS successfully updated the record.

  HELD MANTIS failed to update the record because it was held by another user. This situation normally calls for a finite number of retries of the GET statement, since individual records would not normally be held for long.

- MANTIS may also return a status of NOTFOUND, LOCK, or ERROR if TRAP is in effect.

  NOTFOUND The requested key does not exist.

  LOCK The password specified in the ULTRA statement is invalid.

  ERROR An error occurred while updating the record.

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

Example

```
20 ULTRA BILL("BOM","ASSEMBLY",8)
 .
 .
 .
100 UPDATE BILL LEVEL=LEVEL_NUMBER
```

UPDATE (RDM user view)

NOTE

This statement applies to SUPRA RDM users only.

UPDATE *view-name* [LEVEL=*level-number*]

view-name

Description	*Required*. Specifies the symbolic name of the RDM user view that you want to update.
Format	MANTIS symbolic name as defined in a previously executed VIEW statement

LEVEL=*level-number*

Description	*Optional*. Specifies which level of MANTIS array elements contains the updated values of the row contents.
Default	LEVEL=1
Format	Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding VIEW statement
Consideration	You must specify LEVEL=*level-number* if the corresponding VIEW statement specifies *levels* unless you want the default value.

General considerations

- ♦ Since RDM rows may not be uniquely keyed, you should establish your current row position in a view (by reading the row) before you execute an UPDATE.
- ♦ MANTIS obtains the value of each field in the RDM user view from the corresponding MANTIS variable or array element and writes it to the row to be updated.

- You can obtain the status of the UPDATE by using the “view-name” built-in text function. The values are GOOD and HELD.

  GOOD   MANTIS successfully updated the row.

  HELD   MANTIS failed to update the row because it was held by another user.

- MANTIS may also return a status of NOTFOUND or ERROR if TRAP is in effect.

  NOTFOUND   The requested key does not exist.

  ERROR   An error occurred while updating the row. Something may be wrong with the database, or you may have tried to perform an invalid function on the view.

- RDM returns three status indicators that indicate processing results. You can obtain these indicators using the built-in functions FSI, ASI, and VSI. FSI indicates the success or failure of the update. ASI indicates the status of each field in the row. VSI indicates the highest field status within the row. For a complete discussion of these status indicators, see “DBCS support feature” on page 455.

- If you perform an UPDATE without a previous GET...ENQUEUE, the system internally performs a GET...ENQUEUE prior to performing the UPDATE.

- Use the GET...ENQUEUE prior to using the UPDATE function when computing a value for a row based on the current value, such as incrementing a counter. If you are simply using the UPDATE to place a new value in a row, it is not necessary to issue a GET...ENQUEUE statement.

- You cannot update a view key. By altering the view key, you have actually requested a repositioning of the view, not a modification of the current row. To update a view key, you must first delete the old row and then insert a new one.

- If TRAP is not in effect and the update cannot be applied because of a failure status from RDM, MANTIS automatically issues a RESET. If TRAP is in effect and the program does not issue a RESET when ERROR is returned, it is possible that MANTIS will do only part of the modification.

- Your database administrator may disallow updates. If so, MANTIS returns the ERROR status if TRAP is in effect. Check the FSI message for the reason. If TRAP is not in effect, MANTIS displays a message and halts execution.
- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

Example

```
 10 VIEW CUSTOMER("CUST")
 20 SCREEN MAP("CUST_UPDATE")
 30 SHOW"ENTER CUSTOMER NUMBER:";
 40 OBTAIN CUST_NO
 50 GET CUSTOMER(CUST_NO)
 60 IF CUSTOMER="FOUND"
 70 .INPUT_OK=FALSE
 80 .UNTIL INPUT_OK
 90 ..CONVERSE MAP
100 ...
110 ...  Edit input data here
120 ...
130 ..WHEN EDIT_OK
140 ...INPUT_OK=TRUE
150 ..END
160 .END
170 .UPDATE CUSTOMER
180 .SHOW"CUSTOMER INFORMATION UPDATED"
190 ELSE
200 .SHOW"CUSTOMER NOT FOUND"
210 END
```

UPPERCASE

Use the UPPERCASE function to return the specified text value with any lowercase letters changed to uppercase.

UPPERCASE(*t*)

t

Description *Required*. Specifies the text expression you wish to convert to uppercase.

Format Specifies any valid text expression

Consideration See also “LOWERCASE” on page 280.

Example

```
SHOW UPPERCASE ("Lend Me Your Ears")
LEND ME YOUR EARS
```

USER

Use the USER function to return a 1–16 character text value containing the user name entered during MANTIS sign-on.

USER

General consideration

- ♦ See also “PASSWORD” on page 299.

Example

```
 10 ENTRY LIST_ENTITIES
 20
 .
 .
 .
 80 .FILE REC(ENTITY,PASSWORD,4)
 90 .HEAD"ENTITY"+ENTITY+"FOR USER"+USER
100 .GET REC FIRST LEVEL=1
```

USERWORDS

Use the USERWORDS function to return the number of MANTIS symbolic names currently in use.

USERWORDS

The USERWORDS function is included for IBM compatibility.

Example

```
SHOW USERWORDS
```

VALUE

Use the VALUE function to return the numeric value of a text expression.

$$\textbf{VALUE(}\boldsymbol{t}\left[\begin{array}{c}\textbf{,BIG}\\ \textbf{,DECIMAL}\\ \textbf{,INTEGER}\end{array}\right]\textbf{)}$$

t

Description	*Required*. Specifies a text expression to convert to a numeric value.
Format	Text expression

BIG, DECIMAL, INTEGER

Description	*Optional* Specifies the numeric data type of the return value of the function. The data type has a bearing on how the text string is interpreted and may affect the accuracy of expression results.
Default	BIG

General considerations

- MANTIS ignores all nonnumeric characters except for decimal point (.), plus sign (+), minus sign (-), and exponent indicator (E).
- MANTIS evaluates the text expression using the numeric considerations found in "Numeric considerations" on page 51.
- If INTEGER is specified, only the digits in the text string are processed.
- If DECIMAL is specified, E-notation is not recognized.
- If BIG is specified or no *type* is specified, D-notation is not recognized.
- See also "FORMAT" on page 217 and "TXT" on page 363.

Example

```
40 .DO BROWSE_RTN
50 .CONVERSE MAP
60 .WHILE MAP<>"CANCEL"
70 ..WHEN VALUE(CMD(1,5))>ZERO AND VALUE(CMD(1,5))<99999
80 ...CODE=VALUE(CMD(1,5))
90 ...DO REPOINT(MAP,REC,CODE)
```

```
SHOW VALUE("123.5", INTEGER)
1235
SHOW VALUE("123e5", DECIMAL), VALUE("123e5")
1235         12300000
```

VIEW

Use the VIEW statement to define an RDM user view your program accesses. MANTIS opens a user view consisting of the fields you select in a specified view. MANTIS obtains the attributes of each field in the user view from the database directory and defines a MANTIS variable or array with the same name (except that hyphens (-) in field names are converted to underscores (_) in MANTIS variable names). VIEW can also be used to sign on to or sign off from RDM.

NOTE

This statement applies to SUPRA RDM users only.

$$\textbf{VIEW}\ \left\{\begin{array}{l} \textit{view-name}(\textit{view}[,\textbf{PREFIX}][,\textit{levels}][\textbf{,SELECT}(\textit{field-list},\ldots)]) \\ \textbf{ON}\,[(\textit{user-name}[,\textit{password}])] \\ \textbf{OFF} \end{array}\right\},\ldots$$

view-name

Description *Optional*. Specifies the symbolic name of the RDM user view in subsequent GET, UPDATE, INSERT, DELETE, TRAP, and MARK statements.

Format Unique MANTIS symbolic name

Considerations

- No processing takes place if the *view-name* has already been defined.
- You must supply a view-name if you do not select the ON or OFF options.

view

Description *Optional*. Specifies the name of the view as defined in the database directory.

Format Text expression that evaluates to the name of the view in the database directory

PREFIX

Description *Optional*. Indicates that MANTIS place the prefix “view-name_” on all variable names associated with this view. For example, if the user view BOTTLES has a field named VOLUME, the following statement causes the corresponding variable BIN_VOLUME to be defined in the work area: 10 VIEW BIN(“BOTTLES,”PREFIX)

levels

Description *Optional*. Specifies the number of levels you want to use in GET, UPDATE, INSERT, or DELETE statements for this RDM user view.

Format Arithmetic expression that evaluates to a value in the range 1–255

Considerations

- If you specify *levels*, you must also specify LEVEL=*level-number* (unless you want the default LEVEL=1) in all GET, UPDATE, INSERT, DELETE, and MARK statements for that user view.
- If you do not specify *levels*, MANTIS defines an INTEGER, SMALL, BIG, DECIMAL, or TEXT variable for each field in the user view, according to the type defined in the database directory.
- If you specify *levels*, MANTIS defines a one-dimensional array for each field instead of a variable to allow a separate value to be stored at each level. The dimension of each array is *levels*.

SELECT(*field-list*,...)

Description *Optional*. Specifies one or more fields in the logical view to be included in the RDM user view.

Format Each list is a text expression that contains the logical view field names separated by a comma (,). Each field name must conform to naming conventions established by RDM and to all restrictions specified in this document.

Considerations

- MANTIS converts any underscores (_) to hyphens (-) in the requests made to RDM.
- This parameter is translated to uppercase upon execution of your program.

ON

Description *Optional*. Causes MANTIS to sign on to RDM.

Consideration If you have not used VIEW ON with a user name and password, MANTIS signs on to RDM with your MANTIS user ID and password. In this case, your MANTIS user name must be defined in the database directory with the same password as your MANTIS password. (Only the first eight characters of the MANTIS password, forced to uppercase, are used to access the directory.)

user-name

Description *Optional*. Specifies the user name for RDM sign on.

Format Text expression that evaluates to a user name defined in the database directory

Consideration If you do not specify a user name, MANTIS uses the user name specified in a preceding VIEW ON statement or the MANTIS user name if there is no preceding VIEW ON statement.

password

Description *Optional*. Specifies the password for RDM sign on.

Format Text expression that evaluates to the password defined in the database directory for the specified user

Consideration If you do not provide a password, MANTIS uses the password specified in a preceding VIEW ON statement or the MANTIS password if there is no preceding VIEW ON statement.

OFF

Description *Optional*. Causes MANTIS to sign off from RDM.

General considerations

- If you fail to include required fields in the RDM user view when using the SELECT option, you cannot perform inserts and some updates.
- Modifying the key order with SELECT(*field-list*) could adversely affect performance. Your database administrator has defined the key order in the database directory to maximize performance.

- The characters $ and # are invalid in MANTIS variable names. If the view has field names with these characters in them, MANTIS returns an error message and halts execution.
- All logical views used by your program must refer to the same database description and the compiled database description must exist as XXXXXX.MOD, where XXXXXX is the name of the database description. You must assign the name XXXXXX to the logical name CSI_SCHEMA. If the compiled database description does not reside in your default directory, you must assign its file specification to the logical name XXXXXX.MOD. ULTRA 1.5 users must define the logical name as ULTSCHEMA.
- If you want MANTIS to sign on to RDM with a user name other than your MANTIS user name, specify the user name (and corresponding password) in a VIEW ON statement prior to any VIEW statement which defines a user view.
- The user name used to sign on to RDM must also be defined in the database directory as an authorized user of the views specified in VIEW statements.
- Use a VIEW OFF statement before calling a MANTIS interface program if the interface program requires that your process is not signed on to RDM. You can then use VIEW ON to sign back on to RDM after control is returned from the interface program, although MANTIS does this automatically if necessary on the next VIEW statement or the next GET, UPDATE, INSERT, or DELETE for a user view.
- You should need to use VIEW ON or VIEW OFF only in special circumstances.

Examples

```
10 VIEW CUSTOMER("CUST")
20 VIEW CUST_ITEM("CUST_ITEM",10,SELECT("CUST_NO",
30 '"ITEM_NUM","QUANTITY_ON_ORDER"))
60 INTERFACE SPECIAL("RDMSPECIAL","RDMRDMRDM")
70 VIEW OFF
80 CALL SPECIAL
90 VIEW ON
```

view-name

Use the view-name function to return a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on an RDM user view with the symbolic name “view-name.”

This statement applies to SUPRA RDM users only.

view-name

view-name

Description *Required*. Specifies the symbolic name of an RDM user view.

Format MANTIS symbolic name as defined in a previously executed VIEW statement

Example

```
10 VIEW CUSTOMER("CUST")
  .
  .
  .
SHOW CUSTOMER
```

VSI( )

Use the VSI function to indicate the most adverse field status within a row.

This statement applies to SUPRA RDM users only.

VSI(*view-name*)

view-name

Description	*Required*. Specifies the symbolic name by which you refer to the RDM user view.
Format	MANTIS symbolic name defined in a previously executed VIEW statement.

General considerations

- Validity Status Indicators (VSIs) reflect the overall validity of the RDM user view row you used in your last request. See “DBCS support feature” on page 455 for more details.
- See also “ASI (statement)” on page 107, “FSI” on page 220, and “view-name” on page 395.

Example

```
 20 VIEW PARTS("PARTS_ON_ORDER")
  .
  .
  .
100 GET PARTS
110 IF VSI(PARTS)="CHANGED"
  .
  .
  .
  5 VIEW ON("EXAMPLES","CASINO")
 10 VIEW CUSTOMER("CUST")
  .
  .
  .
990 VIEW OFF
```

WAIT

Use the WAIT statement to suspend execution of your program until you press RETURN or a reply key sequence. Use a WAIT statement to prevent data displayed by SHOW from being overwritten by a subsequent statement before you have had time to read it.

WAIT

General considerations

- ♦ PROMPT and CONVERSE completely overwrite the screen as does STOP if it returns to a selection menu. Executing a WAIT before these statements delays their execution until you indicate (by pressing RETURN or a reply key sequence) that you have read any information previously displayed by SHOW.
- ♦ If you execute an OBTAIN statement to obtain input data between SHOW and PROMPT, CONVERSE or STOP, a WAIT statement is unnecessary.
- ♦ See also “OBTAIN” on page 290 and “SHOW” on page 334.

Example

```
10 FILE REC("PERSONNEL", PASSWORD)
20 GET REC
30 WHILE REC <> "END"
40 .SHOW SURNAME",";FIRSTNAME
50 .GET REC
60 END
70 SHOW "Press RETURN to continue...";
80 WAIT
90 CHAIN "MASTER:START_FACILITY"
```

WHEN-END

Use the WHEN-END statement to execute a block of statements when a specified condition is met. MANTIS performs the test before executing the block of statements. If the condition is false, execution drops through to the next WHEN or to the END statement.

```
WHEN expression
.
statements
.
[ WHEN expression ]
[ .               ]
[ statements      ]
[ .               ]
  END
```

expression

Description *Required*. Specifies the condition which must be true for MANTIS to execute the following block of statements.

Format Expression which evaluates to TRUE (non-zero) or FALSE(zero)

General considerations

- After a TRUE condition, execution continues with the next WHEN statement. MANTIS evaluates every WHEN statement unless a BREAK statement is used. A BREAK statement causes execution to pass to the statement after the END.
- See also “BREAK” on page 146, “IF-ELSE-END” on page 249, and “NEXT” on page 286.

Example

```
 10 ENTRY INDEX
 20 .FILE RECORD("INDEX","SERENDIPITY")
 30 .SCREEN MAP("INDEX")
 40 .GET RECORD
 50 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
 60 ..CONVERSE MAP
 70 ..WHEN MAP="PF1"
 80 ...INSERT RECORD
 90 ..WHEN MAP="PF2"
100 ...DELETE RECORD
110 ..WHEN MAP="PF3"
120 ...UPDATE RECORD
130 ..END
140 ..GET RECORD
150 .END
160 EXIT
```

WHILE-END

Use the WHILE-END statement to execute a block of statements repeatedly while a specified condition is true. If the condition is FALSE, MANTIS terminates the WHILE and executes the statement after END. If the condition is TRUE, MANTIS executes the statements within the WHILE range and then reevaluates the condition.

WHILE *expression*

.

statements

.

END

expression

Description *Required*. Specifies the condition which must be true while MANTIS executes the block of statements.

Format Expression which evaluates to TRUE (non-zero) or FALSE (zero)

General considerations

- WHILE-END may not execute the enclosed statements even once.
- Both the WHILE and the END statements must appear on a line by themselves. Any additional statements coded on the end of a WHILE or an END statement (and separated by a colon) are ignored by MANTIS.
- The body (statements) will not be executed at all if the expression is initially FALSE (zero). If you need execute the block once regardless of the initial condition, use the UNTIL-END statement.
- See also “FOR-END” on page 214, “IF-ELSE-END” on page 249, “UNTIL-END” on page 373, and “WHEN-END” on page 399.

Example

```
 10 ENTRY INDEX
 20 .FILE RECORD("INDEX","SERENDIPITY")
 30 .SCREEN MAP("INDEX")
 40 .GET RECORD
 50 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
 60 ..CONVERSE MAP
 70 ..WHEN MAP="PF1"
 80 ...INSERT RECORD
 90 ..WHEN MAP="PF2"
100 ...DELETE RECORD
110 ..WHEN MAP="PF3"
120 ...UPDATE RECORD
130 ..END
140 ..GET RECORD
150 .END
160 EXIT
```

ZERO

Use the ZERO function to return the value zero.

ZERO

General considerations

- See also “FALSE” on page 209, “NOT” on page 288, “SGN” on page 333, and “TRUE” on page 362.

Example

```
10 ENTRY NEW_EXAMPLE
20 .FILE RECORD("JACKSON","CASINO",PREFIX)
30 .TEXT STATE(8),COUNTY(12),TOWN(12)
40 .HEAD "POPULATION REPORT":SHOW " "
50 .CLEAR
60 .GET RECORD
70 .NATION_COUNTER=ZERO
.
```

4

IBM compatibility considerations

There is a high degree of compatibility between MANTIS and MANTIS for the IBM mainframe. You can share application development between minicomputer and mainframe environments. Using MANTIS, you can develop screens, MANTIS files, programs, and other entities, which you can transfer to MANTIS for the IBM mainframe using the Universal Export Facility. MANTIS Release 2.8 is compatible with MANTIS for the IBM mainframe. Note that the Universal Export Facility was known as the IMPORT/EXPORT Facility in MANTIS Release 2.4. Entity transfer to MANTIS for the IBM mainframe in MANTIS Releases 2.1 to 2.3 was achieved using the Migrate Utility (no longer supported).

Some differences between MANTIS and MANTIS for the IBM mainframe result from the internal architecture of the two products and the different hardware on which they run. For example, in the non-IBM environment, MANTIS allows you to have bigger programs with more variables than MANTIS for the IBM mainframe and is free of the constraints on screen layout imposed by the IBM 3270 display format.

MANTIS applications that you create in the non-IBM environment and that you intend to export to the IBM environment cannot take advantage of some of the features available with MANTIS. Such applications must be confined to the non-IBM environment features that are strictly compatible with the MANTIS for the IBM mainframe environment. MANTIS provides an “IBM compatibility mode” option (IBM MANTIS Option) to tell MANTIS to alert you each time you use a feature that is not IBM compatible.

NOTE

All paths of a program must be executed to ensure that it is IBM compatible.

This chapter describes the MANTIS features that are affected by IBM compatibility mode or need special consideration in applications that are to be exported to the MANTIS for the IBM mainframe environment. This table applies to MANTIS for the Mainframe, release 5.4.01 or higher.

Size limits

Various size limits can be specified by MANTIS Options. When IBM compatibility mode is enabled (for example, SET $OPTION “IBM”) the lower MANTIS for the IBM mainframe limits are automatically set in the appropriate MANTIS Options. You can override these Option settings if you wish using SET $OPTION as a command, but for true compatibility this is not recommended.

	MANTIS environments		
Size limit	**MANTIS option**	**OpenVMS**	**IBM**
Maximum string length	MAXSTRLEN	255–65535	255
Maximum dimension size	MAXDIMSIZE	255–65535	255
Maximum number of dimensions	MAXNODIMS	2–255	2
Maximum program line number	MAXPROGLINE	9999–64000	30,000
Maximum user word number	MAXVARS	2047–65535	2047
Maximum number of defined users	MAXUSERS	255–65535	255
Maximum number of external DO levels	MAXDOLEVELS	5–255	5
Maximum number of CHAIN parameters	MAXCHARGS	40–255	40
Maximum program size	MAXPROGSIZE	32768–999999	65535

Array dimensions

The string maximum length subscript is counted in an array dimension count. Therefore, if the MAXNODIMS MANTIS Option is 2, TEXT(15,4,16) is not allowed.

Auto-mapped BIG to SMALL

MANTIS for the IBM mainframe *does not* permit BIG variables defined by ACCESS, INTERFACE, ULTRA or VIEW statements to auto-map onto existing SMALL variables, but it *does* permit this for FILE statements. According to the dissimilarity fault checking rules, MANTIS should fault on this condition because the precision of SMALL is less than that of BIG. However, for compatibility with old releases of MANTIS, no such fault occurs unless the IBM option is turned on and an ACCESS, INTERFACE, ULTRA or VIEW statement attempts the illegal auto-map.

Indirect reference

Use of the ampersand symbol (&) to make an indirect reference to a MANTIS symbolic name is not allowed in IBM compatibility mode.

Field validation routines

Execution of Field Validation Routines is not allowed in IBM compatibility mode.

ACCESS statement

The FILE, NEW and REPLACE parameters are not allowed in IBM compatibility mode. If the external file design's maximum record length is greater than 18000, the ACCESS statement will fault. The External File Design Facility prevents you from creating such profiles when run in IBM compatibility mode.

ATTRIBUTE statement

The PRINTER parameter is allowed in IBM compatibility mode, but in the IBM environment, use of this parameter may be restricted to the MANTIS Master User.

The DETECTABLE, NON DETECTABLE, OVERLINE, NO OVERLINE, LEFT BAR, NO LEFT BAR, RIGHT BAR, NO RIGHT BAR, MODIFIED, and UNMODIFIED attributes are ignored in MANTIS, but take effect in programs which are exported to MANTIS for the IBM mainframe.

The C*nn* field attributes for color are not supported for the ATTRIBUTE statement when in compatibility mode.

The LOWERCASE and UPPERCASE attributes are ignored in programs which are exported to MANTIS for the IBM mainframe, Release 4.2. MANTIS for the IBM mainframe Release 5.2 supports the LOWERCASE and UPPERCASE attributes.

The following attributes cause a program fault in IBM compatibility mode if they are specified in the ATTRIBUTE statement:

- UNDERLINE FULL FIELD
- REVERSE FULL FIELD
- PROTECTED INPUT ONLY
- MESSAGE

The CLASS attribute has a different interpretation in non-IBM and IBM environments.

BIND command

You can export bound programs to or from either MANTIS system, but the associated binding information is discarded. The IBMCHECKS and IBMPCIFAULT MANTIS Options control what action MANTIS takes when you BIND programs or attempt to execute bound programs that contain IBM incompatibilities, while in IBM compatibility mode. By default, MANTIS performs the following checks in IBM compatibility mode:

- ♦ If you attempt to execute a bound program containing IBM incompatibilities, MANTIS issues an error message to inform you that the program is not IBM compatible.
- ♦ When you BIND a program, MANTIS checks for syntactical incompatibilities with MANTIS for the IBM mainframe, and issues a warning message for each incompatibility encountered.

For more information about the IBMCHECKS and IBMPCIFAULT MANTIS Options, refer to *AD/Advantage MANTIS Administration OpenVMS/UNIX*, P39-1320.

CHAIN statement

For IBM compatibility, the COMCHAIN and RELCHAIN MANTIS Options must be set.

The number of arguments passed on the CHAIN statement must be either zero or exactly equal to the number of formal arguments on the ENTRY statement of the chained-to program.

CONVERSE statement

In MANTIS for the IBM mainframe, the WINDOW parameter selects window mode in which the program function keys are used for windowing functions instead of their normal purpose. Window mode is terminated by the completion of CONVERSE processing or by use of PF9/21.

Windowing in MANTIS is restricted to the size of the current map set, that is, you can scroll only to the edge of the map set. Windowing in MANTIS for the IBM mainframe is restricted to the size of the logical display, that is, you can scroll to anywhere in the logical display area, regardless of the map set size.

In MANTIS, there is no window mode. Windowing functions can be performed on the auxiliary keypad whenever it is in function keypad mode (the default). Function keypad mode remains in effect until numeric keypad mode is selected using GOLD/N.

In MANTIS, the only difference between the WINDOW and DISPLAY parameters is that the window position map is displayed if WINDOW is specified.

The template parameter is not allowed in IBM compatibility mode. The RELEASE parameter is not allowed in IBM compatibility mode. The TIME parameter is not allowed in IBM compatibility mode.

CURSOR function

In IBM compatibility mode, the parameters “INDEX1” and “INDEX2” are not recognized. Furthermore, the parameters “FIELD” and “SCREEN” must be uppercase in IBM compatibility mode.

DATE statement

In IBM compatibility mode, the DATE statement parameter is a text expression of 0–12 characters which is converted to uppercase upon program execution. Furthermore, the escape character (%) is not recognized and is treated as a punctuation character. The special substitution string MMM (month name) is also not recognized. The DATE statement is not supported in MANTIS for the IBM mainframe Release 4.2, but is supported in Release 5.2.

DECIMAL restrictions

In MANTIS for the IBM mainframe, DECIMAL is not supported. However, external fields (for example on an External File) may be decimal; they are converted to BIG or SMALL in MANTIS.

DEQUEUE statement

The ultra-name parameter is ignored in MANTIS, but takes effect in programs which are exported to MANTIS for the IBM mainframe.

DISPLAY statement

In IBM compatibility mode, the DISPLAY statement is not allowed in MANTIS programs, but may be used as a command in the Program Design Facility.

DO statement

Arguments which are expressions are not allowed in IBM compatibility mode. For external subprograms, arguments must be defined in IBM mode. In an IBM environment, you may have an undefined parameter on a DO statement. It defaults to BIG. If you have an undefined parameter on a DO statement for an external subroutine, an error message is issued.

EDIT LIST statement

The EDIT LIST statement is not allowed in IBM compatibility mode.

EDITRCV LIST statement

The EDITRCV LIST statement is not allowed in IBM compatibility mode.

GET statement

In IBM compatibility mode, the NEXT parameter is not allowed when a key is also specified for MANTIS files or external files.

$LOGICAL function

Use of the $LOGICAL statement/function is not allowed in IBM compatibility mode.

$OPTION function

In IBM compatibility mode, the $OPTION function is not allowed in MANTIS programs, but may be used in a command (for example, SHOW $OPTION ("IBM")). The SET $OPTION statement may be used only as a command in IBM compatibility mode.

OUTPUT statement

In the IBM environment, MANTIS gives control to a printer exit if the VIA exit parameter is used. In a non-IBM environment, MANTIS ignores the VIA exit parameter.

PERFORM statement

In MANTIS, the parameter of the PERFORM statement is a system command which can perform any function, including running a program. In MANTIS for the IBM mainframe, the parameter of the PERFORM statement is the name of the program or transaction to be performed (with other optional information).

PRINTER statement

In a non-IBM environment, the parameter of the PRINTER statement is either the device name of the printer (or equivalent logical name) or a print file name to which output is spooled. In the IBM environment, the parameter of the PRINTER statement is a CICS or VTAM identifier. These parameters are not meaningful in exported programs unless both systems have been configured with identically named printers. This could be achieved using logical names on the system. It is recommended, however, that applications to be exported avoid use of the PRINTER statement and send their printed output to the default printer device in the user profile.

PROMPT statement

Interactive help libraries are not supported in MANTIS for the IBM mainframe. The optional second parameter to the PROMPT statement is not allowed in compatibility mode.

RELEASE statement

The access-name parameter is not allowed in IBM compatibility mode.

SCROLL statement

In MANTIS for the IBM mainframe, the ON and OFF parameters select the SCROLL mode of the terminal; but, in MANTIS, these parameters are ignored.

SET statement

The SET statement is not allowed in IBM compatibility mode.

SHOW statement

In IBM compatibility mode, data items must be separated by a comma or semicolon. When not in IBM compatibility mode, the separator can be omitted to display the data items without intervening spaces.

In IBM compatibility mode, a comma or semicolon must be used as a separator after ‘AT column,’ but it has no effect on the column position. When not in IBM compatibility mode, a comma or semicolon after ‘AT column’ advances to the next screen zone or column position.

SLICE and SLOT statements

In the IBM environment, the parameters of these statements have an effect on system performance, but in a non-IBM environment they do not.

STATS statement

The STATS statement is not allowed in IBM compatibility mode.

$SYMBOL function

The $SYMBOL function is not allowed in IBM compatibility mode.

TIME statement

In IBM compatibility mode, the TIME statement parameter is a text expression of 0–12 characters which is converted to uppercase upon program execution. Furthermore, the escape character (%) is not recognized and is treated as a punctuation character. The TIME statement is not supported in MANTIS for the IBM mainframe Release 4.2, but is supported in Release 5.2.

TXT function

The TXT0 MANTIS Option must be set to a space for IBM compatibility.

ULTRA statement

The ON and OFF clauses of the ULTRA statement are not allowed in IBM compatibility mode.

Screen design

MANTIS for the IBM mainframe does not support line-drawing characters in a screen design. When MANTIS for the IBM mainframe imports a screen design containing box fields, the box fields are translated to heading fields composed of “+,” “-” and “!” characters to simulate line drawing characters.

MANTIS for the IBM mainframe does not support the FIELD SENSITIVE VALIDATION, MESSAGE and PROTECTED INPUT ONLY attributes.

In MANTIS, the COLOR attribute can be set to BLUE, RED, GREEN, TURQUOISE, NEUTRAL, YELLOW, or PINK. For more information about mapped colors, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300. The Cnn color specification is not IBM compatible. Fields with such specifications are assigned the attribute NO COLOR when the screen is imported from MANTIS to MANTIS for the IBM mainframe.

Field height and width specifications are not IBM compatible and are discarded when a screen is imported by MANTIS for the IBM mainframe.

The DETECTABLE, NON DETECTABLE, OVERLINE, NO OVERLINE, LEFT BAR, NO LEFT BAR, RIGHT BAR, NO RIGHT BAR, MODIFIED, and UNMODIFIED attributes are ignored in MANTIS, but take effect in programs which are imported by MANTIS for the IBM mainframe.

The Valid Name, Validation List, Validation Variable and Field Exit Routine extended edit attributes in Screen Design are not IBM compatible.

The Field Entry Routing attribute is not supported in MANTIS for the IBM mainframe.

MANTIS for the IBM mainframe does not support Automatic Windowing (which is the default in MANTIS). Specify N (no) for this attribute in Library Functions if you want the windowing to be like MANTIS for the IBM mainframe. However, when Automatic Windowing is disabled, you cannot update a window-clipped field on screen, even when the map is conversed with the UPDATE option. This is not compatible with MANTIS for the IBM mainframe.

In the IBM environment, MANTIS requires that every field be preceded by a field separator. Do not specify N (no) for the FIELD SEPARATORS attribute in the Library Facility if the screen is to be exported to the IBM environment.

MANTIS for the IBM mainframe does not support floating maps. Release 4.2 of MANTIS for the IBM mainframe does not support opaque maps, however they are supported in Release 5.2.

External file design

In the IBM environment, MANTIS does not support:

- Alternate keys through the external file view.
- Mailboxes.
- Passwords for SHARED UPDATE access.
- A MAXIMUM RECORD SIZE greater than 18000 bytes.

ASCII and EBCDIC collating sequences

In MANTIS, text values are stored using the ASCII code for each character, whereas MANTIS for the IBM mainframe uses the EBCDIC code for each character. The different numerical values of ASCII and EBCDIC codes give rise to different collating sequences. Comparison of text values can therefore give different results in MANTIS and MANTIS for the IBM mainframe, except when testing for equality.

5

Programming techniques

This chapter offers suggestions for using the MANTIS Logical Terminal Interface and external DO.

MANTIS Logical Terminal Interface

MANTIS performs terminal I/O through a Logical Terminal Interface that supports a logical display area of 32767 rows by 32767 columns. This display area enables you to design and converse screens (maps) that are larger than your physical screen. A single screen design can be as large as 255 rows by 255 columns. Your physical screen acts as a moveable window on the logical display, and you can scroll around the logical display by using the keys described in “Introduction” on page 21.

NOTE

Windowing in MANTIS is relative to the size of the current map set, that is, you can scroll only to the edge of the map set. (Windowing in MANTIS for the IBM mainframe is restricted to the size of the logical display, that is, you can scroll to anywhere in the logical display area, regardless of the map set size.)

The MANTIS Screen Design Facility enables you to design and save screens for use in your application. The MANTIS Program Design Facility enables you to build programs that send your screen designs to physical devices (such as a physical terminal and printer).

Building a map set in your program

In your applications, you can add individual maps to the logical display to form a map set. Maps within a map set are displayed according to their entry sequence into the map set as specified in programming statements and overlay each other. The last non-floating map added to the map set is known as the active map. Every field on this map that falls within the current window is displayed. All other maps in the map set are passive maps.

A map's position within the map set is controlled by the first set of parameters on the CONVERSE statement:

```
                        ┌                 ┌      ┐ ┌┌         ┐┐                          ┐
                        │                 │WAIT  │ ││ WINDOW  ││                          │
                        │ [(row1,col1)]   │SET   │ ││ DISPLAY ││                          │
                        │                 │UPDATE│ └└         ┘┘                          │
                        │                 └      ┘                                      │
CONVERSE screen - name  │                                                                │
                        │                                                                │
                        │    [(row2,col2)]      ┐[template] [TIME = timeout]              │
                        │                       ┘                                        │
                        │ RELEASE                                                        │
                        └                                                                ┘
```

The WAIT, SET, and UPDATE options on the CONVERSE statement determine whether a physical I/O occurs (whether the map set is displayed) and whether the displayed input fields on the passive maps are protected or unprotected. Maps can be conversed over any part of a currently conversed screen, including over headings or data fields. You can create error message maps or a single map into which you write different text in different situations. Because maps are translucent by default, all fields not completely overlaid by the active map show through the active map. When conversing maps using the UPDATE option of the CONVERSE statement, the following rule governs whether you can update partially displayed fields on a passive map. Fields on a passive map that are partially displayed because they are overlaid by the active map cannot be updated; fields on a passive map that are partially displayed because they overlap the physical screen boundary can be updated as long as the AUTOMATIC WINDOWING attribute is specified for the active map. Note that fields with the PROTECTED or PROTECTED INPUT ONLY attribute are always protected against data entry update, irrespective of whether they are partially displayed or whether the UPDATE option has been specified.

Let’s take another look at the CONVERSE statement, this time with the WAIT, SET, and UPDATE options:

$$\textbf{CONVERSE}\ \textit{screen - name}\left[\begin{array}{l} [(\textit{row}1,\textit{col}1)]\left[\begin{array}{l}\textbf{WAIT}\\ \textbf{SET}\\ \textbf{UPDATE}\end{array}\right]\left[\left\{\begin{array}{l}\textbf{WINDOW}\\ \textbf{DISPLAY}\end{array}\right\}\right. \\ \qquad [(\textit{row}2,\textit{col}2)]\ \left.\right]\ [\textit{template}]\ \textbf{[TIME = timeout]} \\ \textbf{RELEASE}\end{array}\right]$$

- CONVERSE *screen-name* without a WAIT, SET, or UPDATE creates a new map set. This map set contains only the named screen and causes a physical I/O to occur, thereby displaying the map on your terminal. The row/column coordinates (the default is row 1, column 1) specify the row and column positions for a screen.

- CONVERSE *screen-name* WAIT adds the named map to the current map set but does not cause a physical I/O to occur. The named map is not displayed.

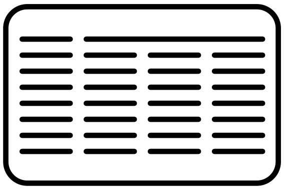
- CONVERSE *screen-name* SET adds the named map to the current map set and causes a physical I/O to occur. This is the active map in the map set, that is, it appears on top of the other maps. This means overlapping fields from the active map have display precedence. All data fields in underlying maps are protected.

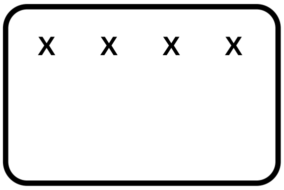
- CONVERSE *screen-name* UPDATE acts the same as CONVERSE *screen-name* SET, except all data entry fields that are fully displayed from underlying maps are unprotected, provided they don’t have either the PROTECTED or PROTECTED INPUT ONLY attribute. Unprotected fields that extend beyond the boundaries of the physical screen do not need to be completely displayed to be updated, provided the AUTOMATIC WINDOWING attribute is specified for the active map.

The final map display can consist of a single map or a composite set of maps positioned dynamically within the logical display. If the map display is larger than the physical screen, the operator can treat the physical screen as a window, moving it around the logical display by using windowing keys (see “MANTIS terminal functions” on page 32).

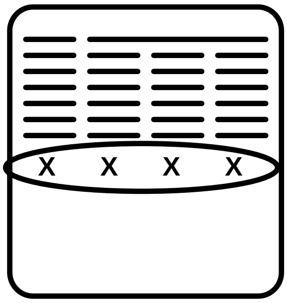
CONVERSE

Using the following screen designs, we will see how one screen overlays the other:





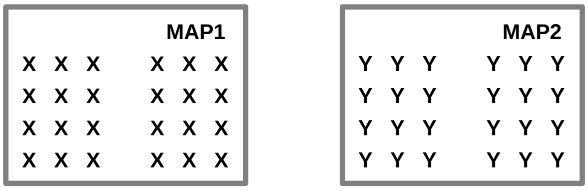
When you converse the maps into one display, you obtain the following results:



```
CONVERSE REPORT WAIT
CONVERSE SUBTOTALS (CO+1,1) SET
```

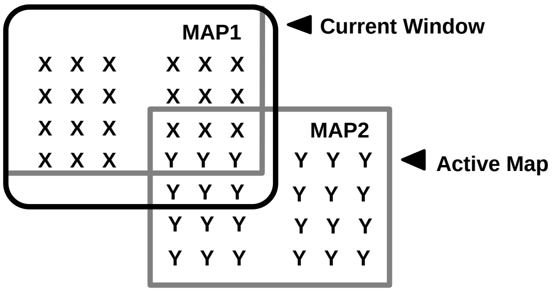
(where CO represents the number of data lines before the subtotal)

The following maps and program discuss more options of the CONVERSE statement, looking at active and passive maps:



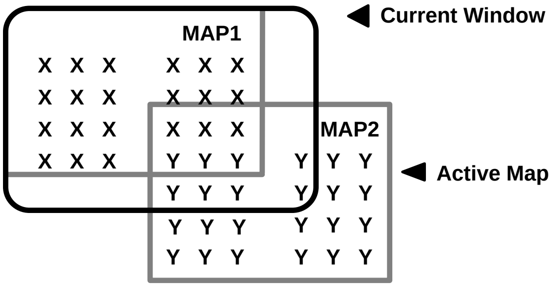
```
 10 ENTRY CLIENT_ENTRY
 20 .SCREEN MAP1("NEW_CLIENT")
 30 .SCREEN MAP2("PAGE_2")
 .
 .
 .
 70 .CONVERSE MAP1                (Adds MAP1 to the map set and
  causes
 .                                 the physical I/O of the map set to
  the
 .                                 screen.)
 .
120 ...IF FIELD="Y"
160 ....DO EXTRA_INFO
 .
 .
 .
300 ENTRY EXTRA_INFO
310 .UNTIL MAP2="CANCEL" OR MESSAGE=""
320 ..CONVERSE MAP2(10,45) SET
 .
 .
 .
400 .IF MAP2="ENTER"
410 ..INSERT REC
420 ..CLEAR
```

The CONVERSE statement in line 320 produces the following result. Note the fully displayed fields on MAP1 are protected because of the SET parameter used in the CONVERSE statement. Also, the last two fields on MAP1 are overlaid by the first two fields on MAP2.

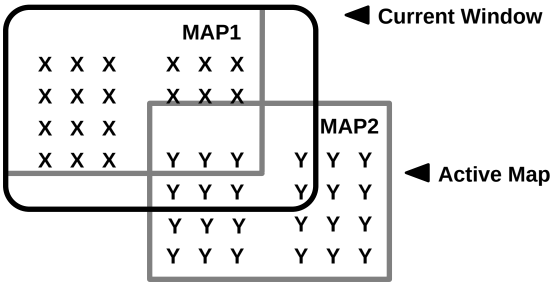


Below is a variation on the CONVERSE statement in line 320. This variation uses the UPDATE option instead of the SET option. Note that the fully displayed input fields on MAP1 are unprotected:

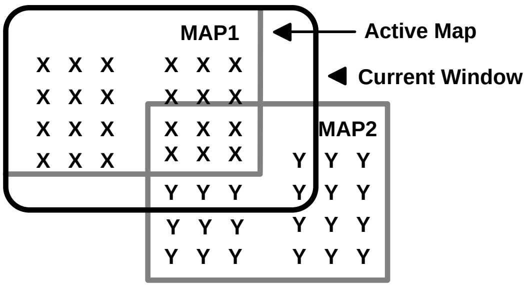
```
CONVERSE MAP2(10,45) UPDATE
```



You can create an opaque map by using the Opaque option in the Library Functions of Screen Design. The active opaque map completely overlays the parts of the passive map it covers.

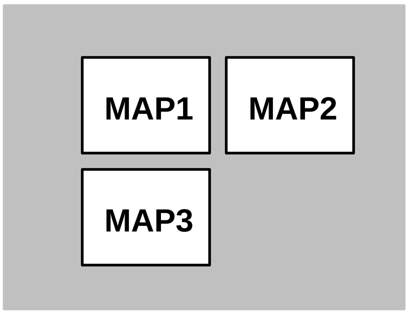


Executing a CONVERSE MAP1 UPDATE again produces the following result:

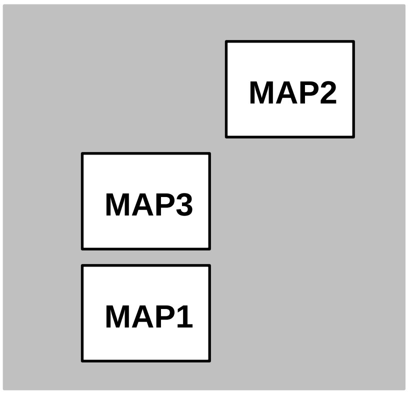


Multiple images of a single screen design

You can converse a screen only once in a map set. If you converse a screen a second time and specify row and column coordinates different from those in the first CONVERSE statement, the screen moves to the new location within the map set. In the following example, the CONVERSE statement in line 210 moves MAP1 from its original position (1,1) to its new position (60,1):



```
110  CLEAR
120  CONVERSE MAP2 (1,80)
130  CONVERSE MAP1 WAIT
140  CONVERSE MAP3 (30,1) SET
  .
  .
  .
```



```
210  CONVERSE MAP1(60,1) WAIT
```

Note that the CONVERSE in line 120 causes a physical I/O. The map remains in the map set.

If you want one screen to appear multiple times within one logical display, you must define a new SCREEN variable for it each time you want it to appear. In the following example, MAP1 and MAP2 are identical:

```
10 SCREEN MAP1("NAME"), MAP2("NAME")
 .
 .
 .
100 CONVERSE MAP1 WAIT
110 CONVERSE MAP2(45,1) WAIT
120 CONVERSE MAP3(15,1) SET WINDOW (15,1)
```

Each field on a screen has a unique name and value. If you are using the same screen design more than once in your logical display, you will probably want to assign different values to a field that appears in each screen occurrence. To do this, you must prefix each successive occurrence of the screen as follows:

```
100 SCREEN MAP1(NAME,PREFIX)
```

In the above example, MAP1 is the symbolic name for the Screen Design called NAME. All variables for this screen are prefixed with this symbolic name and an underscore (“MAP1_”).

For every occurrence of the same screen on the logical display use a separate symbolic name. All but the first occurrence of the Screen Design must be prefixed if the field names need to be unique.

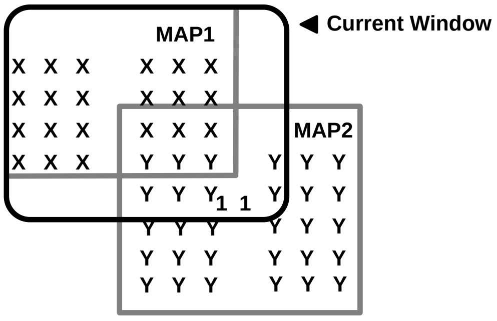
Windowing

You can extend the CONVERSE statement to determine the physical display position of the map set. Because the logical display is 32767 columns wide and 32767 rows long and you can design a screen as large as 255 rows by 255 columns, your map or map set may overrun the physical screen boundaries. By using the DISPLAY or WINDOW parameter of the CONVERSE statement, you can specify whether the row and column coordinates are displayed and where the physical display is located in the logical display:

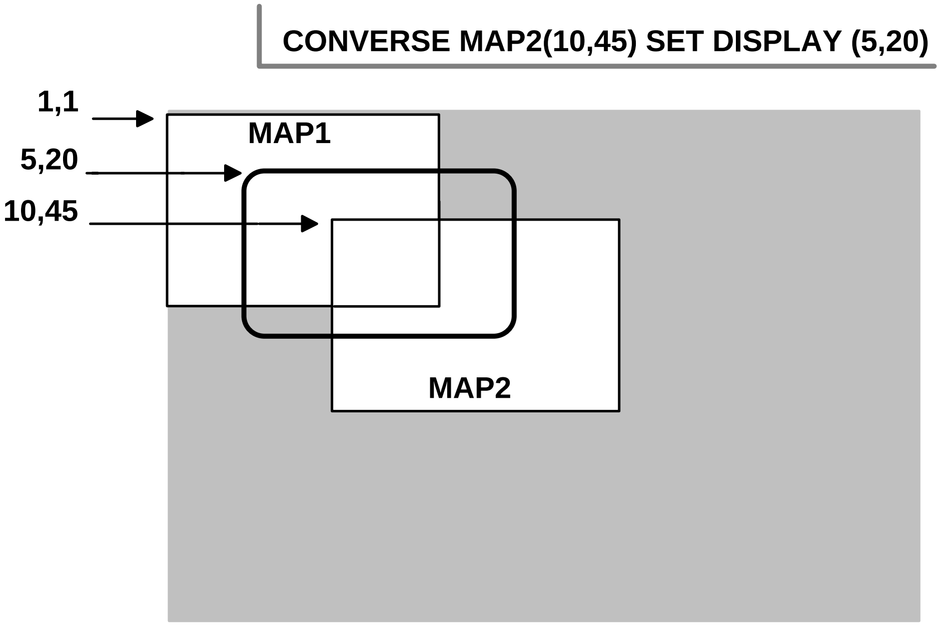
$$
\textbf{CONVERSE}\ \textit{screen - name}
\begin{bmatrix}
[(\textit{row}1,\textit{col}1)]\begin{bmatrix}\textbf{WAIT}\\ \textbf{SET}\\ \textbf{UPDATE}\end{bmatrix}\left[\begin{Bmatrix}\textbf{WINDOW}\\ \textbf{DISPLAY}\end{Bmatrix}\right] \\
\\
[(\textit{row}2,\textit{col}2)]\ ]\ [\textit{template}]\ \textbf{[TIME = timeout]} \\
\textbf{RELEASE}
\end{bmatrix}
$$

CONVERSE *screen-name* WINDOW(*row*,*col*) displays the map with the upper left corner of the window positioned at the specified row and column coordinates of the logical display. These coordinates are displayed at the bottom of the screen, and can optionally be displayed by pressing WINDOWMAP (GOLD/W). Sample displays are shown below. Note the fully displayed input fields on MAP1 are protected with the SET option and active with the UPDATE option:

```
CONVERSE MAP1 WAIT
 CONVERSE MAP2(10,45) SET WINDOW
 CONVERSE MAP2(10,45) UPDATE WINDOW
```



CONVERSE *screen-name* DISPLAY(*row*,*col*) enables you to position the physical screen at the supplied location and does not display the window coordinates.



All windowing occurs while the MANTIS program executes a single CONVERSE statement. You can reposition the physical screen by overtyping the row and column coordinates with:

- New values and pressing ENTER. For example: enter 20 40 to scroll to row 20, column 40.
- Displacement values (+ or - in the first character position and the displacement value in the following character positions). For example, enter +10 -20 to scroll down 10 rows and left 20 columns.
- Incremental values (i in the first position and the incremental value in the following position). For example, enter i20 i80 to scroll down 20 rows (WINDOWN) or up 20 rows (WINUP) and right 80 columns (WINRIGHT) or left 80 columns (WINLEFT).

You can modify window-scrolling amounts within your program by using the SCROLL statement (for example, SCROLL 30,30).

The application program cannot restrict the positioning of the window by the operator. So, the operator can override the key scrolling amounts specified in the program. The program can choose meaningful SCROLL increments for the initial scroll values.

The application program can also use the DISPLAY (row, column) parameters to position the physical screen somewhere other than row 1, column 1 (upper left corner) of the logical display. DISPLAY does not display the row and column window coordinates.

When you move the window, your screen is updated to show the contents of the logical display at the new window position. You may notice, however, that some parts of your screen remain unchanged. For compatibility with MANTIS on the IBM mainframe, remember that MANTIS uses a field separator to simulate the invisible attribute byte that precedes every field displayed on a mainframe terminal. The position in front of every field on a map is reserved, by default, for the field separator. When the field separator of a visible field (at the left of the screen) falls outside the window, MANTIS uses column one for the invisible field separator. Consequently, column one of the screen is always blank for maps designed with field separators. For maps designed without the field separator option, fields on a map are visible in column one.

Conversing a map using a timeout period or no input

The TIME clause of the CONVERSE statement is used for specifying either a timeout period for the input phase, or no input with a delay period.

Use a positive timeout value to specify the number of seconds to wait, after the last keystroke, before the input phase of the CONVERSE is terminated.

A negative or zero timeout value achieves a “display only” CONVERSE—no input is performed. For a negative timeout value, the absolute value specifies the number of seconds to delay after the CONVERSE is performed.

A CONVERSE terminated due to timeout (including the case when TIME<=0) is indicated by the value “TIMOUT” being returned in the screen-name built-in to the MANTIS program.

Clearing a map set

The mapset is cleared of all maps in any of the following events.

- The CLEAR statement with no parameters is executed.
- A CONVERSE statement without WAIT, SET or UPDATE is executed.
- An external subprogram which put a map into the mapset executes an EXIT or RETURN statement.

Maps can be removed from the mapset individually with the CONVERSE map RELEASE statement. External subprograms which CONVERSE maps into the mapset should execute this statement for every such map if the calling program expects the mapset to exist when it regains control. For example, a main program may CONVERSE a single data entry panel and call external subprograms to perform validation of some fields. If an external subprogram converses a pop-up help or error screen, then it must RELEASE that map from the mapset before returning to the main program, otherwise the mapset will be cleared upon return to the main program and the user will see a blank screen.

A CLEAR statement without parameters clears all conversed maps from the map set and sets the KEY function to “CLEAR.” CLEAR with a map name clears data from the named map’s data fields and sets the key value of the named map to null. It does not remove the named map from your map set. CLEAR is useful if it is not convenient to initialize a mapset in the usual way (CONVERSE without WAIT, SET or UPDATE). For example:

```
100 WHILE KEY<>"CANCEL"
110 .CLEAR
120 .WHEN NAME="Y"
130 ..CONVERSE MAP_NAME WAIT
140 .WHEN OPTION="Y"
150 ..CONVERSE MAP_OPTION WAIT
160 .WHEN REFERENCE="Y"
170 ..CONVERSE MAP_REFERENCE WAIT
180 .END
190 .CONVERSE MAP_SELECT SET
200 END
```

The CONVERSE statements (lines 130, 150, and 170) require no considerations if they are the first statements executed. CLEAR (line 110) clears any previous map set so these conversed maps do not overlay an existing map. The CONVERSE statement in line 190 sends the map to the logical display.

External DO

An external DO statement enables a MANTIS program to transfer data and control of execution to another MANTIS program and return to the next statement in the original program. To use external DO efficiently, follow the guidelines discussed here when designing applications.

External DO works similarly to internal DO except that the subroutine resides externally to the program containing the DO statement. Any program can use both internal and external DOs. Internal routines are defined on ENTRY statements within the original program. External routines are defined in PROGRAM statements within the original program.

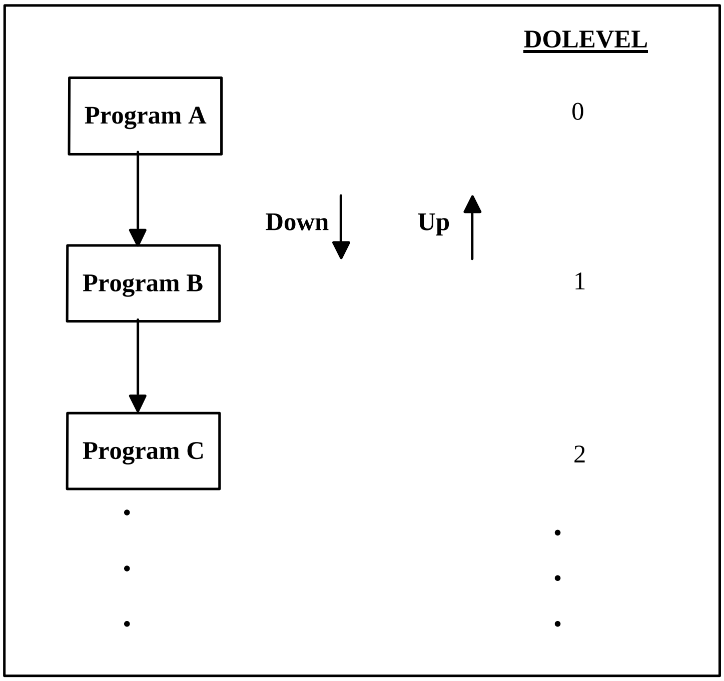
Internal DO	enables a MANTIS program to transfer data and control of execution to a routine within the same program and return to the next statement following the DO.
CHAIN	enables a MANTIS program to transfer data and control of execution to another MANTIS program without an automatic return path.
CALL	allows a MANTIS program to transfer data and control of execution to a non-MANTIS program with a return to the program statement following the CALL.
PERFORM	enables a MANTIS program to invoke a system command.

External DO can improve the modularity of your program and make it easier to maintain your applications by enabling many programs to share common subroutines. The following keywords support external DO:

PROGRAM	identifies the external MANTIS programs to be invoked.

DOLEVEL returns a value identifying which level an external program is currently executing. The original program has a DOLEVEL of zero. With each external DO statement, the DOLEVEL value increases by one. With each exit from an external subroutine, the DOLEVEL decreases by one.

DOWN and UP enable you to list and edit an external subroutine in the work area. DOWN selects an external subroutine at a lower level. UP selects an external subroutine at a higher level. The following figure shows DOLEVELs in relation to the DOWN and UP commands.



For more information about the UP and DOWN commands, refer to *AD/Advantage MANTIS Facilities OpenVMS/UNIX*, P39-1300.

Passing parameters

In IBM compatibility mode, MANTIS passes parameters by passing data names by reference only, and not the actual data. User-defined symbolic names for entities can be passed. In addition, before using a data name in a DO statement, you must define it using BIG, SMALL, TEXT, LET, SCREEN, FILE, ACCESS, SHOW, OBTAIN, ULTRA, VIEW, INTERFACE, or implicitly by reference in a MANTIS expression.

With MANTIS, if you are not operating in IBM compatibility mode, you may specify a parameter as a symbolic name or an expression. If the parameter is a symbolic name, each reference to the corresponding argument in the subroutine uses the specified data item (MANTIS passes data by reference). If you specify a parameter as an expression, it is evaluated before calling the subroutine and each reference to the corresponding argument in the subroutine uses the value of the expression (MANTIS passes data by value). Note that a subscripted variable is considered to be an expression, and is passed by value.

A maximum of 255 parameters can be passed in one DO statement and only passed parameters can be referenced outside the routine where they are defined. Individual variables defined as part of a SCREEN, FILE, ACCESS or other entity definition statement must be passed as separate parameters. They are not passed by just passing the symbolic name for the entity (SCREEN, FILE, ACCESS).

Data referenced by parameters are global to both the original routine and the subroutine. For symbolic names, modified values are retained on return to the original program. For expressions that are passed by value, values assigned to the corresponding ENTRY arguments are lost upon return from the subroutine. Parameter passing can affect automatic mapping.

The names of the arguments passed in the originating program's DO statement need not match the names of the parameters in the external program's ENTRY statement. The following ENTRY statement accepts MAP1, MAP2, FIELD1, and FIELD2 as M1, M2, F1, and F2:

```
DO PROG2(MAP1,FIELD1,MAP2,FIELD2)
.
.
.
ENTRY PROG2(M1,F1,M2,F2)
```

Once you pass the maps, you can converse or clear them and update variables within the external program (PROG2). The passed maps and variables need not be reinitialized with SCREEN or LET statements.

To enter variables into previously existing maps while executing an external DO, you must perform another CONVERSE. To enter data in multiple maps, CONVERSE UPDATE the last conversed map. Or, CONVERSE UPDATE a map defined within the program executed by an external DO over the existing maps.

In the following example, the fields in MAP1 can be updated:

```
ENTRY PROG1
.
.
.
CONVERSE MAP1 WAIT
DO PROG2
.
.
.
ENTRY PROG2
SCREEN MAP2("TWO")
CONVERSE MAP2 UPDATE
```

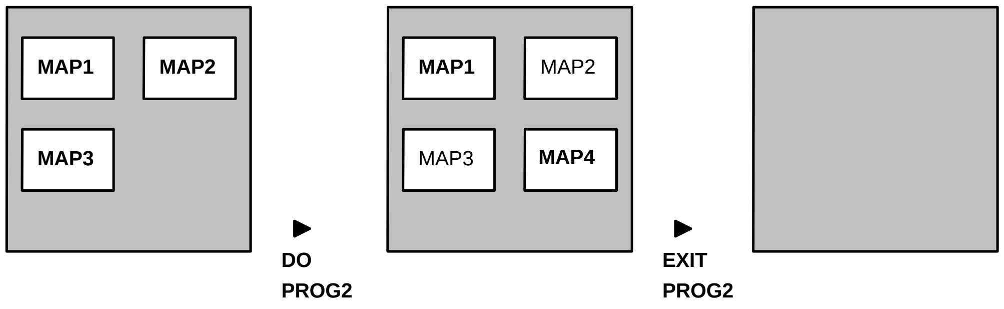
The examples below show how map names are passed to the external program by naming them as arguments of the originating program's DO statement and as parameters of the external program's ENTRY statement. The DO statement in PROG1 passes MAP2 and MAP3 as arguments to PROG2. PROG2 also uses MAP4, which is not defined in PROG1.

In the first example, MAP4 is not cleared before exiting PROG2. Because MAP4 remains in the map set and is not defined in PROG1, the entire map set is cleared when returning to PROG1:

```
ENTRY PROG1
. SCREEN MAP1("ONE")
. SCREEN MAP2("TWO")
. SCREEN MAP3("THREE")
. CONVERSE MAP1 WAIT
. CONVERSE MAP2(1,20) WAIT
. CONVERSE MAP3(20,1) SET
. DO PROG2(MAP2,MAP3)
```

```
ENTRY PROG2(MAP2, MAP3)
. SCREEN MAP4("FOUR")
. ATTRIBUTE(MAP2)="NOR"
. ATTRIBUTE(MAP3)="NOR"
. CONVERSE MAP4(20,20) SET
EXIT
```

(re-enter PROG1 at statement after DO)

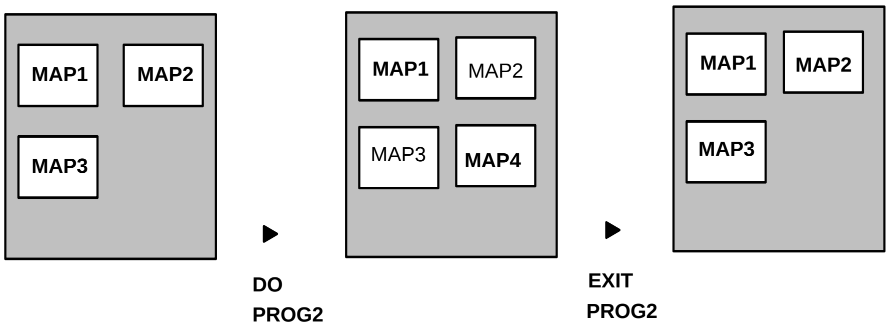


In the next example, PROG2 converses MAP4 and then clears it from the map set. Therefore, the map set, which now contains only maps already defined in PROG1, remains when returning to PROG1:

```
ENTRY PROG1
. SCREEN MAP1("ONE")
. SCREEN MAP2("TWO")
. SCREEN MAP3("THREE")
. CONVERSE MAP1 WAIT
. CONVERSE MAP2(1,20) WAIT
. CONVERSE MAP3(20,1) SET
. DO PROG2(MAP2,MAP3)
```

```
ENTRY PROG2(MAP2, MAP3)
. SCREEN MAP4("FOUR")
. ATTRIBUTE(MAP2)="NOR"
. ATTRIBUTE(MAP3)="NOR"
. CONVERSE MAP4(20,20) SET
. ATTRIBUTE(MAP2)="RES"
. ATTRIBUTE(MAP3)="RES"
. CONVERSE MAP4 RELEASE
EXIT
```

(re-enter PROG1 at statement after DO)



Program architecture

Internally, a MANTIS program consists of three main areas:

- Program Work Area**:** contains the program code.
- Data Work Area**:** contains all the variables used by the program, including complex variables such as ACCESS and SCREEN.
- Vocabulary, or Symbolic Name Table**:** a directory of all variable names used in the program. It allows MANTIS to find the data associated with a variable name, without searching the entire Data Work Area.

All code and data work areas associated with a MANTIS program are collectively referred to as the Program Context Area (PCA). Each DOLEVEL of an active subprogram chain has its own PCA. The MAXDOLEVELS MANTIS Option is provided to safeguard against over utilization of memory through too many nested external DO statements.

When a subprogram returns control to its caller via EXIT or RETURN, the subprogram PCA is saved in memory in a PCA pool local to the calling program. The MAXPCAPOOL MANTIS Option sets the limit on the number of PCAs allowed in a PCA pool at every DOLEVEL.

When a PCA pool becomes full, MANTIS starts replacing pooled PCAs with new PCAs being released by EXIT and RETURN. The oldest pool entry is always selected for replacement on the assumption that the more recently called subprograms are also the more likely ones to be called in the near future—but this is not always the case, as shown in Case B of the following example:

Case A

```
PROGRAM A1,A2
WHILE NOT(FINISHED)
.DO A1
END
WHILE NOT(FINISHED)
.DO A2
END
```

Case B

```
PROGRAM A1,A2
WHILE NOT(FINISHED)
.DO A1
.DO A2
END
```

Case B degenerates into the worst case when the number of subprograms (A1, A2,... A^n) exceeds the value of the MAXPCAPOOL MANTIS Option, because when the PCA for program AN is released into the pool it will replace saved context A1, which is the next one required. After that, A1 will replace A2, and so on. In this case MANTIS has to rebuild each PCA on every DO, loading program code from the MANTIS File (or possibly the Shared Entity Pool), and the memory used by the pooled PCAs is nothing but a wasted resource. So it is important to find an optimum value for MAXPCAPOOL.

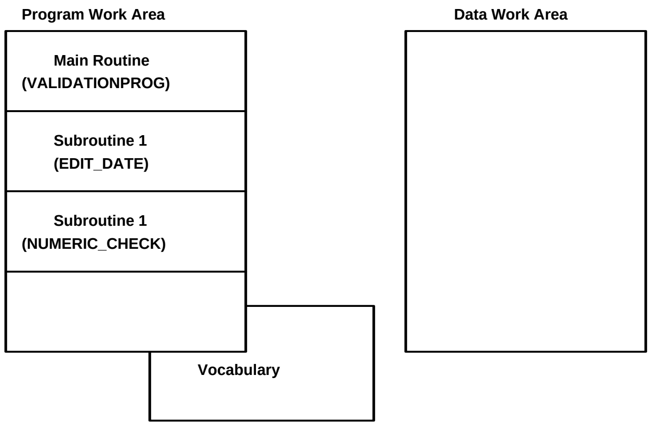
Your application can also control the PCA pool to some extent through the RELEASE statement. Use this statement to force a particular PCA out of the pool when:

- The subprogram is very large and the cost of keeping it in memory is worse than the cost of rebuilding its PCA on each DO.
- The subprogram is infrequently called. By releasing the context from the pool immediately after the DO statement, the oldest pooled PCA is saved from automatic replacement on the next DO.

Internal DO vs. external DO vs. CHAIN

Understanding the characteristics of external DO, internal DO, and CHAIN helps you apply the programming guidelines in the next section. The following figure is an example of an internal DO. This is a mainline routine bound by ENTRY and EXIT statements. The DO statements indicate subroutines that reside in the same program as the calling routine

Execution involves one Program Work Area, one Data Work Area, and one symbolic name table. A maximum of 255 parameters are passed, but all may be global. MANTIS stores the program as one entity.

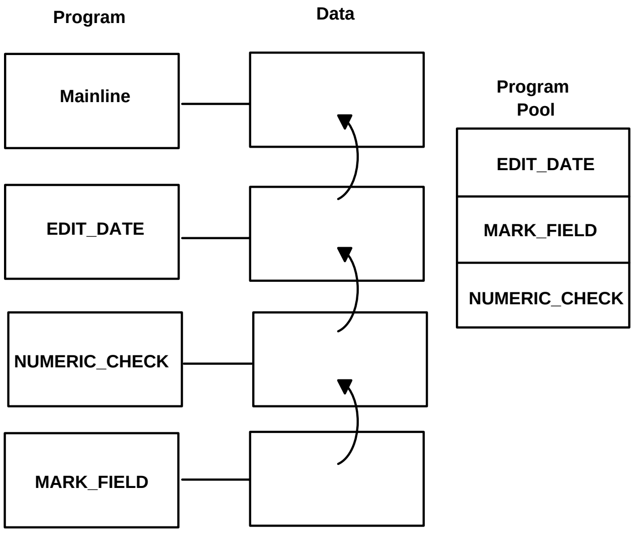


```
ENTRY VALIDATIONPROG
  .
  .
  .
DO EDIT_DATE(MAP...)
  .
  .
  .
EXIT
ENTRY EDIT_DATE(...)
  .
  .
  .
DO NUMERIC_CHECK(...)
  .
  .
  .
EXIT
ENTRY NUMERIC_CHECK(...)
  .
  .
  .
DO MARK_FIELD(...)
  .
  .
  .
EXIT
ENTRY MARK_FIELD(...)
  .
  .
  .
EXIT
```

The following figure is an example of an external DO. You must use ENTRY and EXIT statements around each physical program. (Dash lines indicate physical program boundaries.)

All symbolic names are local to their own program. Therefore, everything the subroutine uses from its caller must be defined on the ENTRY statement. A maximum of 255 parameters may be passed; variables not passed are local to each routine. Each program is stored as a separate entity.

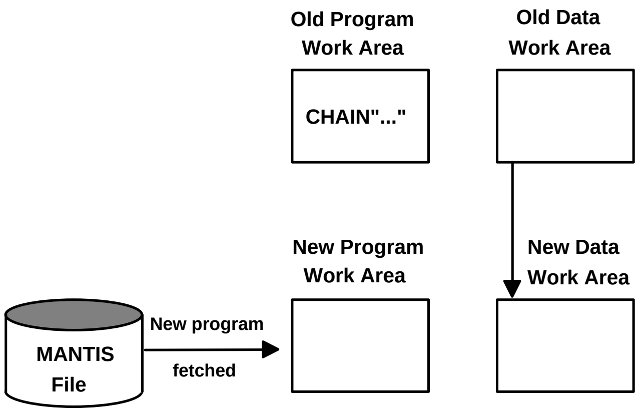
Local Program Chain



```
ENTRY VALIDATEPROG
DO EDIT_DATE(MAP...)
DO MARK_FIELD
EXIT
------------------------
ENTRY EDIT_DATE(...)
DO NUMERIC_CHECK(...)
EXIT
------------------------
ENTRY NUMERIC_CHECK(MAP...)
DO MARK_FIELD(...)
EXIT
------------------------
ENTRY MARK_FIELD(...)
EXIT
```

The following figure illustrates a program using the CHAIN statement. ENTRY and EXIT statements are required around each physical program. (Dash lines indicate physical program boundaries.)

When MANTIS encounters a CHAIN statement, it loads the program from the MANTIS File, the external subprogram pool, or the shared entity pool list, depending on where the program is stored. After loading the program, MANTIS allocates and builds a new Data Work Area and then copies parameters from the old Data Work Area into the new Data Work Area. When it establishes new areas, it releases the old Program and Data Work Areas. The following figure illustrates the sequence when the program resides in the MANTIS file.



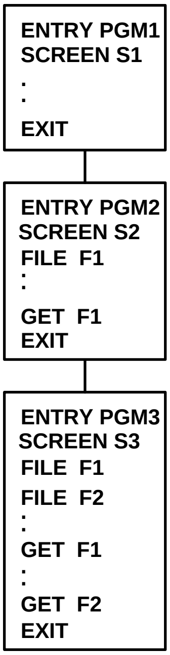
```
ENTRY OLD
   CHAIN "NEW",A,B,C
EXIT
------------------
ENTRY NEW (X,Y,Z)
 .
 .
 .
EXIT
```

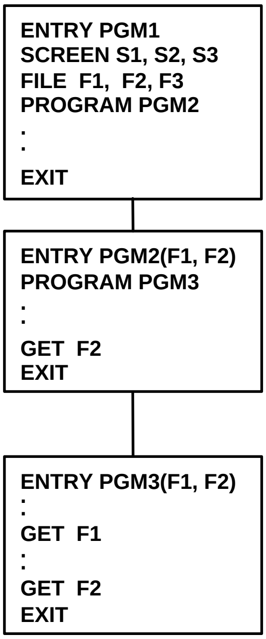
You can use the CHAIN LEVEL statement to indicate that the program at the current DOLEVEL should be overlaid. MANTIS replaces the program at the current DOLEVEL and begins executing the CHAIN program. When the chained program exits, control returns to the caller of the program that performed the CHAIN—as if the program that performed the CHAIN did an EXIT.

Referring to the application in the following figure as an example, let's examine how using various statements affects the number of disk accesses to the MANTIS File. CHAIN links PGM1, PGM2, and PGM3, and each time MANTIS encounters CHAIN it fetches the new program from the MANTIS File. Complex variables (SCREEN and FILE) must be defined in each new Data Work Area.

If you use internal DO, you decrease the number of disk accesses because everything you need to execute is loaded once, which saves fetching programs and screens.

External DO (see the figure on the right, below) enables you to define in the main program variables that are used in all routines. When defined in the main program, they can be passed to subsequent routines. For example, files F1, F2, and F3 are defined in PGM1, and can be passed to PGM2 and PGM3 with no disk access required. The only disk accesses required are those to fetch PGM2 and PGM3 the first time they are loaded.

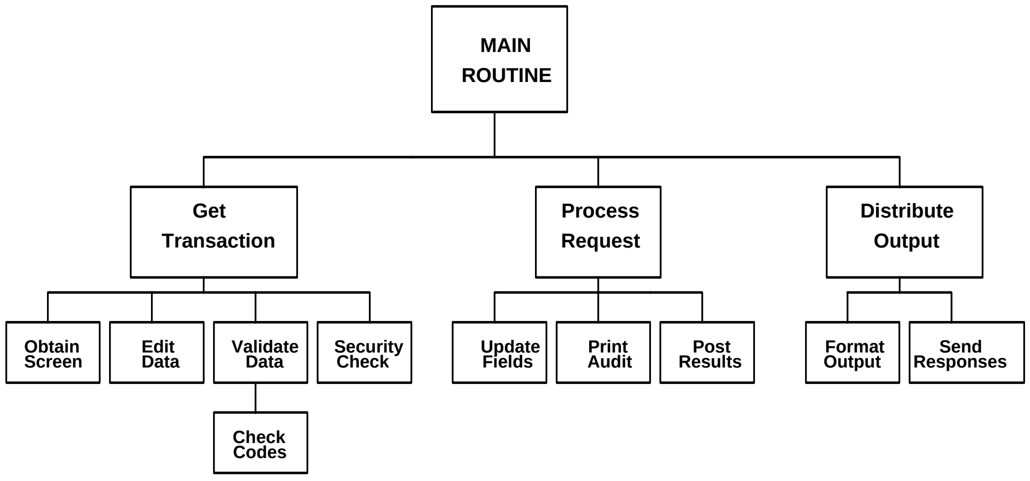




Modularizing

Many small externally done programs produce overhead. Each small program requires extra processing and storage for each DO statement. To execute many small routines, group related routines into one program, then pass a parameter to the program, indicating which routine should be executed. The main routine in the program does internally DO the subroutines.

In a system designed via a structured methodology, you can combine multiple related modules into one program. For example, the following figure shows how you can combine 14 separate modules into three programs. In other words, do not make every module a separate program. Instead, mix internally and externally done routines.



Automatic mapping

Automatic mapping is the process that MANTIS uses to allow the sharing of data areas between variables of like name and data type. Since MANTIS only defines a particular symbolic name one time for a particular program to run, it is possible to use the same variable name on multiple entities (such as SCREENS, FILES and VIEWS), so that MANTIS automatically maps like-named fields from one entity to the other. This mapping occurs automatically whenever an already defined name is encountered. We recommend that you use standard naming conventions on a system-wide basis to make the best use of this feature. In the following example both the FILE and SCREEN have fields defined as F1, F2 and F3. Any information entered into any of these fields is available to both entities (the screen and the file). The CONVERSE can be in either SUB1 or the calling routine, with the same results.

```
10 ENTRY MAINLINE
20 PROGRAM SUB1("SUB1", password)
30 FILE REC("Y","P"):|CONTAINS F1,F2,F3
40 SCREEN MAP("X"):|CONTAINS F1,F2,F3
50 DO SUB1(MAP,REC)
```

This routine automatically maps F1, F2, F3 between MAP and REC

```
10 ENTRY SUB1(SC,FI)
20 .CONVERSE SC
30 .INSERT FI
40 EXIT
```

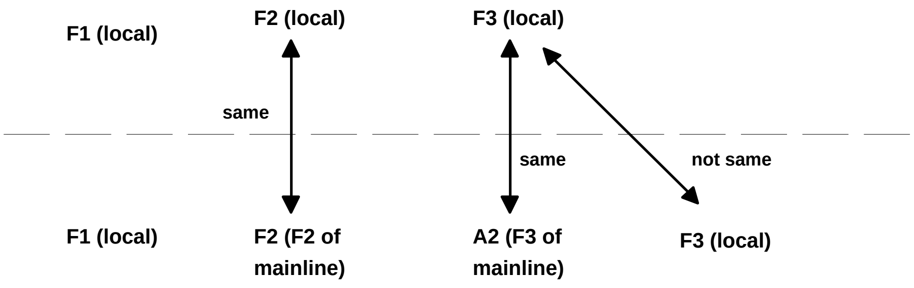
In another example, F1, F2, and F3 are defined in the main routine. On entry to the subroutine, SC, F2, and A2 are defined as arguments in the subroutine. When the FILE statement executes, MANTIS looks for F1 and doesn’t find it, so it defines one. When it encounters F2, it finds an F2 (because it appeared on the ENTRY statement) and connects to it. When MANTIS encounters F3 and doesn’t find it, it defines another F3 (not the one in the main routine). A2 in the subroutine is the local name for the calling routine’s F3 and shares its Data Work Area.

The CONVERSE statement fills in all the variables in SC. The only variable automatically mapped in the CONVERSE and INSERT is F2.

Since F2 is defined as an argument on the ENTRY statement, it is automatically be mapped between the file and the screen. Even though F3 is passed as a parameter, the name “F3” is not known to the subroutine SUB1. Therefore the FILE statement establishes a local variable F3 (as well as F1) in SUB1.

```
10 ENTRY MAINLINE
20 PROGRAM SUB1("SUB1", password)
30 SCREEN MAP("X"):|CONTAINS F1,F2,F3
50 DO SUB1(MAP,F2,F3)
------------------------------------------------------
10 ENTRY SUB1(SC,F2,A2)
20 FILE REC("Y","P"):|CONTAINS F1,F2,F3
30 CONVERSE SC
40 INSERT REC
50 EXIT
```

The diagram below shows what happens to the passed variables.



Entity definition

Avoid defining entities within external programs which causes recurring disk I/O to the MANTIS File and rebuilds the symbolic name table. Declare all complex variables (such as SCREEN, FILE, and ACCESS) as high up as possible in the hierarchy of external DOs. However, keep in mind the constraints for automatic mapping previously mentioned.

The Data Work Area is released on EXIT and reacquired on each DO. If you have complex statements in subroutines, they must be refetched from the MANTIS File (causing disk accesses) and rebuilt (causing processor overhead) each time the program is done. Even if you cannot define these entities at the highest level (DOLEVEL=0), the higher you can place these statements, the better.

Frequency of DOs

The more often your application uses a subroutine, the more advantage you have using DO over a CHAIN statement (with a CHAIN to return) because the calling routine context is maintained, and the called routine does not need to be fetched from the MANTIS File each time. However, if an external routine can be designed to do as much work as possible per call, performance will be better. For example, if a routine can be called once to validate all elements of an array, it is better than calling repetitively for each element to be validated. Very heavily used routines are best performed as internal routines.

Debugging

Using external DO in programming mode enables you to debug interactively. Use the UP and DOWN commands to move between the active DOLEVELs. Observe the “PROGRAM==>” heading at the top of the screen to determine which program you are in.

Use SHOW DOLEVEL to see which level is executing. You can modify and replace programs at any level. The revised version executes in the next run.

Remember that arithmetic and text variables, lists or arrays should correspond in data type between the ENTRY-EXIT and the DO or CHAIN statements.

You can set breakpoints in a program when you save it. If such a program is invoked via external DO when running in Program Design, the breakpoint occurs at the specified line. This is particularly useful when errors occur deep in a call chain. Note that a breakpoint has no effect when a program is run via the Run a Program facility.

Program size

Moderate the size of your programs. Try not to go to either extreme, that is, a few very large programs or many small programs. If you have many large programs occupying memory as part of the hierarchy of external DOs, the amount of storage needed to run MANTIS applications systems may increase.

Programming consideration

With external DO, you can use the same FILE statement to reference many file descriptions because a new FILE statement is created for each call. For example:

Program 1

```
.
.
PROGRAM FILENEW("PGM2","PASSPGM2")
.
.
.
.
FILE_ID=LIBNAMEX
INSERT_LEVEL=INSERT_PASSWORD
DO FILENEW (FILE_ID,INSERT_LEVEL)
.
.
```

Program 2

```
ENTRY PGM2 (FILE_NAME,PSSWD)
.
.
.
FILE F (FILE_NAME,PSSWD)
.
.
.
```

You can change FILE_ID and INSERT_LEVEL in program 1 before any DO statement. MANTIS re-executes the FILE statement for each call. This also works for other complex variable types (such as SCREEN and ACCESS).

Performance

In externally done routines that will not be called again soon, add a RELEASE program-name after the last external DO to free up the space in the external subprogram pool, and thus reduce the amount of data that needs to be processed. Note that the program must contain both PROGRAM and RELEASE statements. If two or more programs have a PROGRAM statement for one program, any of the programs that have the PROGRAM statement can do the RELEASE, although the last program is preferable. Good candidates for this are one-time routines (for example, initialization, or infrequent help) or routines which are very infrequently called (for example, setting profile information, routines only called at certain times, for example at the end of the month or the beginning of the day).

The next external DO call to any routines which have already been released reloads the program onto the local program chain where it resides again until it is explicitly or implicitly released. Appropriate logic for an infrequently used program can be similar to that shown below.

```
IF choice
.PROGRAM INFR(pgmname,pgmpswd)
.DO INFR(p1,p2,p3)
.RELEASE INFR
END
```

A

Status functions

Status functions

SUPRA RDM returns three types of status indicators to MANTIS indicating the success or failure of an RDM GET, UPDATE, INSERT, or DELETE. MANTIS gives you access to these indicators by providing three functions which return corresponding text values. The three status functions are:

- FSI(*view-name*[,*text-name*]). Function Status Indicator

NOTE

Text-name is a TEXT variable, and the maximum length of message text it can receive is 40 characters. However, if the text-name variable is less than 40 characters in length, truncation may occur.

- ASI(*view-name*,*field-name*). Attribute Status Indicator
- VSI(*view-name*). Validity Status Indicator

The FSI indicates the success or failure of the most recent RDM operation on the specified RDM user view. If the optional text-name parameter is supplied, the accompanying text-name from RDM is copied to the specified text variable. The ASI indicates the status of the specified field in the row. The VSI indicates the most adverse field status within the row. Each function is documented in "MANTIS programming language" on page 89. The tables in this appendix list and describe the indicators these functions may return.

FSI value	Meaning
DATA	Data error. The row contains invalid data. Check the ASIs to find the field(s) with the invalid value(s).
ERROR	Major error. Something may be wrong with the database or you may have attempted to perform an invalid function on the user view. Use the FSI message parameter to find the cause of the error.
GOOD	Successful GET or UPDATE.
NOTFOUND	Failure due to an occurrence problem. This may be due to a GET not found or an INSERT duplicate found and can also be returned at the end of the chain of related records.
RESTART	Restart request on COMMIT. For more information on conditions that cause this to be returned refer to the *SUPRA Server PDM Messages and Codes Reference Manual*, P25-0022.
SECURITY	Security. For more information on considerations that cause this to be returned refer to the *SUPRA Server PDM Messages and Codes Reference Manual*, P25-0022.
UNAVAILABLE	Unavailable resources (for example, a held resource).
RESET	RESET recommended. During processing, function modifications were made to the database before MANTIS detected the error condition. Issue a RESET to restore.

Attribute Status Indicators (ASI) reflect the status of each field defined in your RDM user view. There is an ASI indicator for each field in your RDM user view. MANTIS Attribute Status Indicators appear in the following table.

ASI value	Meaning
CHANGED	The field value was changed by another user.
DATA	The value of the field is not a valid value.
MISSING	The field is missing. It has a null value.
NEW	The field exists, but has changed since the most recent access.
SAME	The field exists and was not changed since the most recent access.

For additional information, refer to the *SUPRA Server PDM Messages and Codes Reference Manual*, P25-0022.

There are three ways to use ASIs:

- When you issue a GET command, certain returned fields may not have a value. You can check this status (on unaltered fields) with the ASI.
- If you receive an FSI indicating a data error, you may use the ASI to find which fields have invalid values.
- When you issue an UPDATE command, another user may have updated a field value since your preceding GET. If this is the case, MANTIS does not perform your UPDATE. Use the ASI function to check this status and determine what course of action to take.

Validity Status Indicators (VSI) reflect the overall validity of the RDM user view row you used in your most recent request. For example, if an ASI returns a 'DATA' status, the VSI contains 'DATA'. Check this indicator before checking each ASI. MANTIS Validity Status Indicators appear in the following table.

VSI value	Meaning
CHANGED	No invalid or missing ASIs were returned, but at least one field in the row has changed.
DATA	At least one ASI indicating an invalid value was returned.
MISSING	No invalid ASIs were returned, but at least one missing ASI was returned.
NEW	No invalid or missing ASIs were returned, but at least one data item in the row has changed.
SAME	No invalid, missing, or new occurrences were returned.

For additional information refer to the *SUPRA Server PDM Messages and Codes Reference Manual*, P25-0022.

The VSI enables you to determine if you need any additional processing of ASIs to correct invalid data or to fill in missing values. There is a one-to-one correspondence between ASIs and VSIs. ASIs reflects the status of each field; VSIs reflect the overall status of the RDM user view row.

MANTIS file and external file status functions

The MANTIS and external file statuses are returned when you perform a DELETE, GET, INSERT, or UPDATE. To obtain the FSI codes, use the following function:

FSI (*filename* [*, message*])

This function returns a status such as GOOD, NOTFOUND, SECURITY, or ERROR, depending upon the status of the last I/O operation. The message field may contain an associated system error message if available. The following is a short example showing usage:

```
10 FILE F ("TEST","PSW")
20 TEXT MSG(100)
30 TRAP F ON
40 ID="No. 3":DATA="updated data"
50 UPDATE F
60 SHOW "File Status =";F
70 SHOW "FSI Result  =";FSI(F,MSG)
80 SHOW "FSI Message =";MSG
```

RUN

```
File Status = HELD
FSI Result  =  UNAVAILABLE
FSI Message = %RMS-E-RLK, target record currently locked by
  another stream
```

The following table maps the MANTIS status to the FSI codes you can receive.

MANTIS status	FSI	Description
“”	“GOOD”	Successful DELETE
“”	“GOOD”	Successful INSERT
“”	“GOOD”	Successful UPDATE
“FOUND”	“GOOD”	Successful keyed GET
“NEXT”	“GOOD”	Successful unkeyed GET
“NEXT”	“NOTFOUND”	Keyed GET without EQUAL
“NOTFOUND”	“NOTFOUND”	Keyed GET with EQUAL
“ERROR”	“NOTFOUND”	Attempt to update nonexistent record
“ERROR”	“UNAVAILABLE”	File not available/not found?
“ERROR”	“ERROR”	All other errors
“END”	“NOTFOUND”	Attempt to GET past EOF
“DUPLICATE”	“DUPLICATE”	Duplicate record on INSERT
“DATA”	“DATA”	Conversion error
“LOCK”	“SECURITY”	No privilege for operation
“HELD”	“UNAVAILABLE”	Attempt to GET but record locked
“HELD”	“UNAVAILABLE”	Attempt to DELETE but record locked
“HELD”	“UNAVAILABLE”	Attempt to UPDATE but record locked

B

DBCS support feature

For OpenVMS users only.

For the benefit of Japanese-language users, MANTIS can support the use of DBCS characters. DBCS Support is a product option which must be requested when MANTIS is ordered. This appendix describes the components of the DBCS Support option.

To run MANTIS with DBCS Support, you also need a DBCS terminal and the KANJI/OpenVMS operating system.

KANJI and MIXED data types

You can use DBCS characters (and Kana characters) in MANTIS text data, but not in MANTIS keywords, symbolic names or numeric data. This means that you must use ASCII characters to write your MANTIS programs (except for text literals and comments), but the text data that they process can contain DBCS and Kana characters. MANTIS supports Japanese characters by means of the text data type TEXT and two new text data types, KANJI and MIXED:

TEXT data consists of ASCII and KANA characters in any combination.

- KANJI data consists of KANJI characters only.
- MIXED data consists of ASCII, KANA and KANJI characters in any combination.

In MIXED data, MANTIS uses special shift codes to indicate the start and end of each group of consecutive DBCS characters. MANTIS inserts a shift-in code before the first DBCS character in a group and a shift-out code after the most recent DBCS character in a group. If MIXED data ends with a DBCS character, MANTIS can dispense with the final shift-out code. The shift-in and shift-out codes show up as spaces when MIXED data is displayed or printed.

Entering DBCS and Kana characters from the keyboard

You can enter DBCS characters into any field on the screen which has the KANJI or MIXED attribute. You position the cursor where you want the DBCS characters to be entered and press the SELECT key or the period (.) key on the auxiliary keypad. MANTIS then prompts you to make your selection by entering DBCS characters on the second last line of the screen. When you press RETURN or ENTER, MANTIS inserts the selected DBCS characters at the cursor position.

If the field has the MIXED attribute, MANTIS may need to insert shift-in and shift-out codes which occupy one screen position and are displayed as spaces. If there was already a space before or after the DBCS characters, MANTIS changes it to a shift-in or shift-out code.

Each DBCS character occupies two positions on the screen. When editing, that is, inserting, overtyping or deleting characters in a KANJI field, the cursor may be positioned over either position in a DBCS character to identify that character as the one to be inserted before, overtyped or deleted. The same applies to MIXED field editing except that in overtype mode, the cursor position specifies where the entered character will be positioned and DBCS data to the left or right of the cursor may be erased in order to preserve the overall appearance of the field.

You can enter Kana characters into any field on the screen which has the TEXT or MIXED attribute. You simply position the cursor and key in the characters using the appropriate shift on the keyboard.

KANJI and MIXED field attributes

Field attributes are defined in the screen maps which are added to the logical display. Some maps are provided as part of the MANTIS facilities; others are designed by MANTIS programmers.

In the maps provided by MANTIS, the MIXED field attribute is used in certain input fields to allow you to enter DBCS, Kana or ASCII characters. In Program Design, for example, you enter program statements in the input field on the bottom line of the screen. This field has the MIXED attribute so you can put DBCS literals and comments in your ASCII programs.

In your own maps, you can specify the KANJI or MIXED attribute using the Update Field Specifications function in the Screen Design Facility.

KANJI and MIXED variables

Text data that you enter into screen fields with the TEXT, KANJI or MIXED attribute is stored in MANTIS text variables of the corresponding type. You can use the KANJI and MIXED statements to define KANJI and MIXED variables, just as you use the TEXT statement to define TEXT variables. You can also create arrays of KANJI and MIXED data with these statements.

The description pages for the KANJI and MIXED statements can be found at the end of this appendix.

When processing a SCREEN statement, MANTIS automatically creates a TEXT, KANJI or MIXED variable for each field with the corresponding field attribute in the screen map.

When processing a FILE, ACCESS, ULTRA, or INTERFACE statement, MANTIS automatically creates a TEXT, KANJI, or MIXED variable for each element of the corresponding type in the profile.

KANJI and MIXED literals and comments

All DBCS, Kana and ASCII characters can be included in a text literal enclosed by quotes ("..."). MANTIS determines whether the literal is TEXT, KANJI, or MIXED according to the characters it contains.

All types of characters can be included in a program comment preceded by a vertical bar (|.).

If you precede a comment by an exclamation mark (!... as in MANTIS for the IBM mainframe KANJI comment), MANTIS substitutes a vertical bar for the exclamation mark.

KANJI and MIXED expressions

MANTIS text expressions can contain operands of the same or different text data types. When one text operand is joined to another using the plus (+) operator, MANTIS determines the data type of the result according to the following rules:

♦	TEXT	+	TEXT	=	TEXT
♦	TEXT	+	KANJI	=	MIXED
♦	TEXT	+	MIXED	=	MIXED
♦	KANJI	+	TEXT	=	MIXED
♦	KANJI	+	KANJI	=	KANJI
♦	KANJI	+	MIXED	=	MIXED
♦	MIXED	+	TEXT	=	MIXED
♦	MIXED	+	KANJI	=	MIXED
♦	MIXED	+	MIXED	=	MIXED

When one text operand is removed from another using the minus (-) operator, MANTIS determines the data type of the result according to the following rules:

TEXT	-	TEXT	=	TEXT	
TEXT	-	KANJI	=	TEXT	the data is unchanged
TEXT	-	MIXED	=	TEXT	
KANJI	-	TEXT	=	KANJI	the data is unchanged
KANJI	-	KANJI	=	KANJI	
KANJI	-	MIXED	=	KANJI	
MIXED	-	MIXED	=	MIXED	
MIXED	-	TEXT	=	MIXED	
MIXED	-	KANJI	=	MIXED	

Text data can be assigned to a variable of the same or different text data type. When data is assigned to a text variable, MANTIS adjusts the data, if necessary, according to the following rules:

TEXT	=	TEXT	the data is unchanged
TEXT	=	MIXED	DBCS characters and shift codes are removed
TEXT	=	KANJI	the result is a null text string
KANJI	=	TEXT	the result is a null DBCS string
KANJI	=	KANJI	the data is unchanged
KANJI	=	MIXED	ASCII, Kana characters, shift codes are removed
MIXED	=	TEXT	the data is unchanged
MIXED	=	MIXED	the data is unchanged
MIXED	=	KANJI	shift codes are inserted

KANJI and MIXED substrings

You can use subscripts to refer to a substring of a KANJI or MIXED variable. As with TEXT variables, the subscripts specify the first and last character positions in the string. When specifying character positions for MIXED variables, do not count shift codes. A substring has the same data type as the original string.

Text data can be assigned to each type of text substring according to the assignment rules stated above. Each character of assigned data replaces one character in the substring, regardless of the type of the characters.

Comparison of KANJI and MIXED data

MANTIS relational expressions can contain operands of the same or different text data types. Shift codes are inserted in KANJI data before comparing it with MIXED data. TEXT data can be compared with MIXED data but not with KANJI data. When comparing text operands, a MANTIS program uses either ALPHANUMERIC or LENGTH based comparison (see “Relational expressions” on page 66 for more information on text comparisons). When ALPHANUMERIC comparison is used, MANTIS first compares the text operands up to the shorter operand length, then compares their lengths only if they are equal up to the shorter length. When a LENGTH comparison is used, MANTIS first compares the lengths of the text operands and compares their values only if their lengths are equal.

MANTIS compares text operands of the same size according to their numeric values which depend on the numeric codes used to store each character. MANTIS uses OpenVMS numeric codes for ASCII, Kana and DBCS characters. If either operand contains DBCS characters, the only useful operator is equals (=) because the numeric codes for DBCS characters do not correspond to the standard collating sequence.

KANJI and MIXED data in MANTIS functions

The POINT function can have an argument containing KANJI and MIXED operands. POINT returns the character position at which addition or subtraction would occur.

The SIZE function can have a KANJI or MIXED argument. SIZE returns the number of characters in the argument (excluding shift codes). You may also specify “BYT” as a second argument for the SIZE function; SIZE then returns the size of the argument (TEXT, KANJI or MIXED) in bytes (including shift codes). Note, however, that if you do specify the “BYT” option, the first argument to the function must be a scalar variable. See a description of this function later in this appendix.

The UPPERCASE and LOWERCASE functions can have a MIXED argument. The UPPERCASE/LOWERCASE functions convert ASCII characters to uppercase/lowercase and leave DBCS and Kana characters unchanged.

The NULL function and null string literal (“”) are treated as TEXT, KANJI or MIXED according to the context of their use.

KANJI and MIXED fields in screen design

Using the Create or Update a Screen function, you can insert DBCS and Kana characters in heading fields at any position on the screen. MANTIS splits headings if necessary into separate fields of type TEXT or KANJI.

You create KANJI and MIXED data fields by using hash characters (#) to indicate the positions occupied by the field on the screen. For a KANJI field, you must indicate an even number of positions. You specify the KANJI or MIXED attribute using the Update Field Specifications function.

By default, MANTIS allows KANJI heading fields and data fields to start in either odd- or even-numbered columns. If you select the ALIGN KANJI option in Library Functions, MANTIS aligns all KANJI fields so that they start in odd-numbered columns which makes the screen IBM-compatible. Select this option before you start to design your screen if you want MANTIS to align KANJI heading and data fields as soon as you specify them.

You cannot use DBCS characters in the edit mask of a numeric field.

Within Screen Design, double height and double width field specification is supported for both KANJI and MIXED fields. Since a DBCS character normally occupies two character positions on the display, DBCS characters in a field specified as double width occupy four character positions.

KANJI and MIXED data in file design

You can use the Update Record Layout function of File Design to specify MANTIS file elements with data type KANJI or MIXED. This causes MANTIS to create KANJI or MIXED variables when a FILE statement refers to your file design. You use these variables to transfer KANJI or MIXED data to and from the MANTIS file.

You specify the length of a KANJI element to be the maximum number of DBCS characters you want to store in the element.

You specify the length of a MIXED element to be the maximum number of bytes needed for the element. Allow one byte for each ASCII character, one byte for each Kana character, two bytes for each DBCS character and two bytes for each pair of shift-in and shift-out codes. You can allow extra bytes if you are not sure how many characters of each type will occur.

KANJI and MIXED data in external fields

You can also specify elements with data type KANJI or MIXED in External File View Design, ULTRA File View Design, and Interface Design. Each of these MANTIS facilities handles KANJI and MIXED elements in the same way.

You specify the length of a KANJI element to be the maximum number of bytes needed for the element. Allow two bytes for each DBCS character. When transferring data from a KANJI variable to a KANJI element, MANTIS appends as many trailing DBCS spaces as are needed to fill up the element. When transferring data from a KANJI element to a KANJI variable, MANTIS removes any trailing DBCS spaces.

You specify the length of a MIXED element to be the maximum number of bytes needed for the element. This depends on the external format used to store the MIXED data. You select one of three formats by specifying the SIGN of the element as NONE, BLANK, or CODE.

NONE	No shift-in or shift-out codes are stored before and after groups of DBCS characters. MIXED elements with this format cannot contain any Kana characters. MANTIS removes shift-in and shift-out codes when transferring data from a MIXED variable to a MIXED element and inserts them when transferring data from a MIXED element to a MIXED variable. Allow one byte for each ASCII character and two bytes for each DBCS character.
BLANK	ASCII spaces are stored instead of shift-in and shift-out codes before and after groups of DBCS characters. MIXED elements with this format cannot contain any Kana characters. MANTIS replaces shift-in and shift-out codes with ASCII spaces when transferring data from a MIXED variable to a MIXED element. MANTIS inserts shift-in and shift-out codes in place of spaces when transferring data from a MIXED element to a MIXED variable. Allow one byte for each ASCII character, two bytes for each DBCS character and two bytes for each pair of shift-in and shift-out codes.
CODE	The actual shift-in and shift-out codes are stored before and after groups of DBCS characters. Allow one byte for each ASCII character, one byte for each Kana character, two bytes for each DBCS character and two bytes for each pair of shift-in and shift-out codes.

When transferring data between MIXED variables and elements, MANTIS faults if the data does not fit or contains Kana characters when it should not. If TRAP is in effect, MANTIS returns a status of “TRUNCATE” or “DATA” respectively.

KANJI menus, messages, and prompters

Japanese versions of the MANTIS facility menu screens, MANTIS messages and HELP prompters are available. MANTIS uses the Japanese versions only if your Master User has specified Japanese as the default language in your User Profile.

KANJI

Use the KANJI statement or command to define KANJI variables or arrays of KANJI variables. A KANJI variable is a type of text variable which can contain only DBCS characters.

KANJI *kanji-name*[([*dimension*,...,]*length*)],...

kanji-name

Description *Required*. Define the symbolic name of the KANJI variable or array.

Format MANTIS symbolic name

dimension

Description *Optional*. Specify an array dimension. Use one dimension parameter to specify a one-dimensional array; two dimension parameters to specify a two-dimensional array, and so on.

Format Arithmetic expression that evaluates to a value in the range 1–*max*, where *max* is the value of the MAXDIMSIZE MANTIS Option.

Consideration The maximum number of dimensions you can specify is determined by the value of the MAXNODIMS MANTIS Option. Each dimension specifies the maximum value of the corresponding array subscript.

length

Description	*Optional*. Specify the maximum length (in characters) of a KANJI variable or array element.
Default	16 DBCS characters
Format	Arithmetic expression that evaluates to a value in the range 1–*max*/2, where *max* is the value of the MAXSTRLEN MANTIS Option.

General considerations

- No processing takes place if *kanji-name* has already been defined.
- When you assign data to a KANJI variable, MANTIS truncates any characters which exceed the maximum length of the KANJI variable.
- The current length is initially set to zero.

Examples

```
20 KANJI ALPHA(64,3), BETA(12)
```

Defines a KANJI array ALPHA with 64 3-character elements and a 12-character KANJI variable BETA.

MIXED

Use the MIXED statement or command to define MIXED variables or arrays of MIXED variables. A MIXED variable is a type of text variable which can contain ASCII, Kana, and DBCS characters in any combination. MANTIS uses special shift codes to indicate the start and end of each group of consecutive DBCS characters. MANTIS inserts a shift-in code before the first DBCS character in a group and a shift-out code after the most recent DBCS character in a group.

MIXED *mixed-name*[([*dimension*,...,]*length*)],...

mixed-name

Description	*Required*. Define the symbolic name of the MIXED variable or array.
Format	MANTIS symbolic name

dimension

Description	*Optional*. Specify an array dimension. Use one dimension parameter to specify a one-dimensional array, two dimension parameters to specify a two-dimensional array, and so on.
Format	Arithmetic expression that evaluates to a value in the range 1–*max*, where *max* is the value of the MAXDIMSIZE MANTIS Option.
Consideration	The maximum number of dimensions you can specify is determined by the value of the MAXNODIMS MANTIS Option. Each dimension specifies the maximum value of the corresponding array subscript.

length

Description	*Optional*. Specify the maximum length (in bytes) of a MIXED variable or array element.
Default	16 bytes
Format	Arithmetic expression that evaluates to a value in the range 1–*max*, where *max* is the value of the MAXSTRLEN MANTIS Option.
Consideration	One byte is required for each ASCII character, one byte for each Kana character, two bytes for each DBCS character and two bytes for each pair of shift-in and shift-out codes.

General considerations

- No processing takes place if *mixed-name* has already been defined.
- When you assign data to a MIXED variable, MANTIS truncates any characters which exceed the maximum length of the MIXED variable.
- The current length is initially set to zero.

Example

```
20 MIXED ALPHA(64,20), BETA(12)
```

Defines a MIXED array ALPHA with 64 20-byte elements and a 12-byte MIXED variable BETA.

SIZE

Use an extension of the SIZE function to return the size and dimensions of an expression, variable, or array for DBCS, mixed or text data.

$$\textbf{SIZE}\left(\begin{Bmatrix} \textit{text - scalar - variable} \\ \textit{kanji - scalar - variable} \\ \textit{mixed - scalar - variable} \end{Bmatrix}\left[\textbf{,"BYT"}\right]\right)$$

text-scalar-variable,"BYT"

Description *Optional*. Tell MANTIS to return the size, in bytes, of the text scalar variable whose name you specify.

Format MANTIS symbolic name

kanji-scalar-variable,"BYT"

Description *Optional*. Tell MANTIS to return the size, in bytes, of the DBCS scalar variable whose name you specify.

Format MANTIS symbolic name

mixed-scalar-variable,"BYT"

Description *Optional*. Tell MANTIS to return the size, in bytes, of the mixed scalar variable whose name you specify.

Format MANTIS symbolic name

General consideration

- Note that if you use the “BYT” option, the first argument must be a scalar variable. Any specified subscripts are ignored.

Example

```
SIZE("ALPHA","BYT")
```

Index